Universidade Federal de Santa Catarina
Programa de Pós-Graduação em Engenharia de Produção

Dayna Maria Bortoluzzi

UMA EXTENSÃO À ARQUITETURA DA INTERNET PELA INSERÇÃO DE UMA CAMADA DE SESSÃO

Florianópolis, março de 2005

Dayna Maria Bortoluzzi

UMA EXTENSÃO À ARQUITETURA DA INTERNET PELA INSERÇÃO DE UMA CAMADA DE SESSÃO

Tese submetida ao curso de Pós-Graduação em Engenharia de Produção e Sistemas do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito parcial para obtenção do grau de Doutor em Engenharia de Produção e Sistemas.

Orientadora: Profa. Elizabeth Sueli Specialski, Dra.

Florianópolis, março de 2005

DAYNA MARIA BORTOLUZZI

UMA EXTENSÃO À ARQUITETURA DA INTERNET PELA INSERÇÃO DE UMA CAMADA DE SESSÃO

Esta tese foi julgada adequada para a obtenção do título de Doutor em Engenharia de Produção, e aprovada em sua forma final pelo Programa de Pós Graduação em Engenharia de Produção, da Universidade Federal de Santa Catarina.

Florianópolis, 30 de março de 2005.

_____

Prof. Edson Pacheco Paladini, Dr.
Coordenador do Programa de Pós-graduação

Banca Examinadora:

_____

Prof[a]. Elizabeth Sueli Specialski, Dra.
Orientadora

_____

Prof. Roberto Willrich, Dr.
Membro

_____

Prof. Alejandro Martins Rodriguez, Dr.
Moderador

_____

Prof. Alexandre Moraes Ramos, Dr.
Membro

_____

Prof[a]. Alessandra Schweitzer, Dra.
Examinadora Externa

_____

Prof[a]. Tereza Cristina Melo de Brito Carvalho, Dra.
Examinadora Externa

AGRADECIMENTOS

Agradeço à Profa. Beth, minha grande amiga, pelas orientações, conselhos e motivação.

Aos membros da banca cujos questionamentos e considerações permitiram o direcionamento do trabalho.

Ao Professor Alejandro, pela amizade, oportunidade, artigos pesquisados e motivação.

Aos amigos da UFSC e do VIAS que colaboraram de alguma forma no desenvolvimento e finalização desta tese, em especial, Karla Garcia, Nilson Modro e Rita Broering pelo apoio e torcida.

A minha família, pais, irmãs, irmão e padrinho Onacli Luis Fabrin que sempre me incentivaram a estudar.

Ao Marcelo, Luís e Marina por uma vida maravilhosa.

RESUMO

BORTOLUZZI, Dayna Maria. **Uma extensão à arquitetura da Internet pela inserção de uma camada de sessão**. 2005. Tese (Doutorado em Engenharia de Produção) – Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis, 2005.

Durante um bom tempo pouco se alterou na implementação dos protocolos de transporte e rede (TCP e IP), talvez porque a necessidade de se alterar algo não tivesse surgido ou por se acreditar que a grande maioria das necessidades das aplicações pudesse ser solucionada com a criação de novos protocolos. Isso explica a quantidade de protocolos existentes hoje. Porém, recentemente, a demanda de aplicações multimídia na rede tem crescido substancialmente, impulsionada principalmente pela indústria de entretenimento. Investimentos em pesquisa vêm sendo feitos direcionando esforços na construção de uma rede mais rápida e que atenda às necessidades das novas aplicações. Hoje se tem uma idéia melhor do que se tinha há vinte anos, sobre a diversidade e os requisitos das aplicações da Internet. Alguns protocolos foram replicados dentro das aplicações, gerando o problema de redundância de código. Para resolver este problema, as funcionalidades comuns devem ser separadas e agrupadas, evitando, assim, que as aplicações tenham de implementá-las. Neste contexto, esta tese propõe a criação de uma extensão à arquitetura da Internet, que contemple as funcionalidades de sessão, contribua para a organização da atual pilha TCP/IP e auxilie no desenvolvimento de novas aplicações. Tal extensão adapta alguns conceitos de sessão propostos pelo modelo de referência RM-OSI, para atender às características da arquitetura TCP/IP. Além disso, é proposto um modelo P2P para a distribuição de tráfego multimídia em tempo-real e duas maneiras de categorização do tráfego da Internet.

Palavras-Chave: Aplicações multimídia. Arquitetura Internet. Serviços de sessão.

ABSTRACT

BORTOLUZZI, Dayna Maria. **An extension to the Internet architectures by inserting a session layer**. 2005. Thesis (PhD in Production Engineering) Production Engineering Post-Graduation Program, Federal University of Santa Catarina, Florianópolis, 2005

During a long time there's been a few alteration in implementing the Internet transport and network protocols (TCP and IP), perhaps due the fact there was no need to alter something. Until few years ago, the scientific community believed most applications could be solved by creating new protocols. However, recently, the multimedia applications demand in the network has been growing, mainly propelled by the entertainment industry. Investment in research has been made, guiding efforts towards the construction of a faster network that fulfills the new applications needs. Today we have a better idea about the Internet applications' diversity and requirements, than twenty years ago. Some protocols were replicated inside applications, generating the code redundancy problem. In order to solve this problem, common functionalities must be separated and grouped, to avoid applications need to implement them. In this context, this thesis proposes the creation of an extension in the Internet architecture. This extension contemplates the session functionalities, contributes to organizing the current TCP/IP stack and helps the development of new applications. Some session concepts were adapted from the RM-OSI reference model in order to achieve the TCP/IP architecture needs. Besides, we proposed a P2P model for the real-time multimedia traffic distribution and two ways to categorize Internet traffic.

Key Words: Multimedia applications. Internet Architecture. Session services

LISTA DE FIGURAS

FIGURA 1 – DIFERENTES RELAÇÕES ENTRE CONEXÃO DE SESSÃO E DE TRANSPORTE: (A) CORRESPONDÊNCIA UM A UM; (B) VÁRIAS CONEXÕES DE TRANSPORTE PARA UMA ÚNICA SESSÃO; (C) UMA CONEXÃO DE TRANSPORTE PARA VÁRIAS SESSÕES. FONTE: ADAPTADO DE TANENBAUM (1991). 27
FIGURA 2 – VISUALIZAÇÃO DOS PASSOS DE EXECUÇÃO DO HTTPSESSION 28
FIGURA 3 – COMPOSIÇÃO DO PROTOCOLO SSL. FONTE: ADAPTADO DE THOMAS (2000) 29
FIGURA 4 – ESTABELECIMENTO DE CONEXÃO ENCRIPTADA NO SSL. FONTE: ADAPTADO DE THOMAS (2000) 31
FIGURA 5 – RESTABELECIMENTO DE UMA SESSÃO NO SSL. FONTE: ADAPTADO DE THOMAS (2000) 32
FIGURA 6 – ARQUITETURA DE PROTOCOLOS WAP. FONTE: OMA (2003) 34
FIGURA 7 – FUNCIONAMENTO DE UMA APLICAÇÃO WAP. FONTE: ADAPTADO DE PROENÇA (2003) 35
FIGURA 8 – MODELO DE REFERÊNCIA DO WAP. FONTE: ADAPTADO DE OMA (2003) 36
FIGURA 9 – FUNCIONAMENTO DO SIP. FONTE: ADAPTADO DE CISCO SYSTEMS (2003) 43
FIGURA 10 – IMPLEMENTAÇÃO DOS PROTOCOLOS MULTIMÍDIA: (A) COM CÓDIGO FONTE DENTRO DAS APLICAÇÕES; (B) ACESSO ÀS FUNCIONALIDADES VIA PONTO DE ACESSO DE SERVIÇO 59
FIGURA 11 – TÉCNICA DE BUFFERIZAÇÃO. FONTE: ADAPTADA DE LU (1996) 60
FIGURA 12 – COMPONENTES DA CAMADA DE SESSÃO PROPOSTA 64
FIGURA 13 – ESTABELECIMENTO DE UMA SESSÃO 69
FIGURA 14 – FINALIZAÇÃO DE UMA SESSÃO 71
FIGURA 15 – REPRESENTAÇÃO DA CONEXÃO DE SESSÃO: A) MULTIPONTO (CSMU); B) *MULTICAST* 73
FIGURA 16 – ESTABELECIMENTO DE UMA SESSÃO CSMU 74
FIGURA 17 – ENVIO DE DADOS 76
FIGURA 18 – SERVIÇO DE SINCRONIZAÇÃO 78
FIGURA 19 – GERÊNCIA DE DIÁLOGO 80
FIGURA 20 – AUTÔMATO DE ATIVIDADE 81
FIGURA 21 – FUNCIONAMENTO DO BUFFER DE PLAYER 83
FIGURA 22 – FUNCIONAMENTO DO BUFFER DE PLAYER 85
FIGURA 23 – FUNCIONAMENTO DO CONTROLE DE SESSÃO MULTIPONTO 86
FIGURA 24 – AUTÔMATO DO BUFFER DE PLAYER DISTRIBUÍDO 87
FIGURA 25 – FUNCIONAMENTO DO COMPONENTE AT 89
FIGURA 26 – MECANISMO DE *PASS-THROUGH* 93
FIGURA 27 – MECANISMOS DE FILTROS BASEADOS NO ID_SESSION 100

LISTA DE TABELAS

TABELA 1 – USO DOS PROTOCOLOS DE TRANSPORTE NAS APLICAÇÕES MULTIMÍDIA. DIVIDIDO POR PAÍS E POR DOMÍNIO. FONTE: ADAPTADA DE SRIPANIDKLCHAI, MAGGS E ZHANG (2004) 51

LISTA DE QUADROS

QUADRO 1 – DESCRIÇÃO DA CRIAÇÃO E ANDAMENTO DE UMA SESSÃO EM UMA APLICAÇÃO WEB 28
QUADRO 2 – ESTABELECIMENTO DE CONEXÃO ENCRIPTADA NO SSL 31
QUADRO 3 – COMPARATIVO ENTRE ESTABELECIMENTO DE CONEXÃO (RM-OSI X SSL) 45
QUADRO 4 – IMPLEMENTAÇÕES RTSP. FONTE: ADAPTADA DE: CS AT COLUMBIA UNIVERSITY(2003) 57
QUADRO 5 – REDUNDÂNCIA DE CÓDIGO 58
QUADRO 6 – DESCRIÇÃO DOS COMPONENTES DA CAMADA DE SESSÃO 64
QUADRO 7 – PRIMITIVAS DE SERVIÇO DO COMPONENTE CS 67
QUADRO 8 – PARÂMETROS DE NEGOCIAÇÃO 68
QUADRO 9 – PRIMITIVAS UTILIZADAS NO ENVIO DE DADOS 76
QUADRO 10 – PRIMITIVAS UTILIZADAS NA SINCRONIZAÇÃO 77
QUADRO 11 – PRIMITIVAS DE GERÊNCIA DE DIÁLOGO 79
QUADRO 12 – PRIMITIVAS DE ATIVIDADE 81
QUADRO 13 – PRIMITIVAS DO BUFFER DE PLAYER 84
QUADRO 14 – PRIMITIVAS DO BUFFER DE PLAYER DISTRIBUÍDO 87
QUADRO 15 – PRIMITIVAS DO COMPONENTE AT 89
QUADRO 16 – CATEGORIZAÇÃO DO TRÁFEGO ENTRE ELÁSTICO E INELÁSTICO 90
QUADRO 17 – TIPOS DE CATEGORIAS DE TRÁFEGO 92

ACRÔNIMOS

API ..... Application Programming Interface
ARP ..... Address Resolution Protocol
ARPANET ..... Advanced Research Projects Agency Network
AT ..... Análise de Tráfego
ATM ..... Asynchronous Transfer Mode
BSD ..... Berkeley Software Distribution
CS ..... Controle de Sessão
CSMu ..... Controle de Sessão Multiponto
DHCP ..... Dynamic Host Configuration Protocol
DNS ..... Domain Name Service
DMS ..... Distributed Multimedia Applications
EF ..... Extensões Futuras
FiFo ..... First-In First-Out
FTP ..... File Transfer Protocol
FS ..... Funções de Sessão
FYI ..... For Your Information
GB ..... Gerencia de Buffer
HTTP ..... HyperText Transport Protocol
HTTP ..... HyperText Transport Protocol Secured
IANA ..... Internet Assigned Numbers Authority
IETF ..... Internet Engineering Task Force
IIS ..... Internet Information Services
IP ..... Internet Protocol
IPsec ..... IP Security Protocol
IPv4 ..... Internet Protocol version 4
IPv6 ..... Internet Protocol version 6
ITU-T ..... International Telecomunication Union – Telecom Standardization
ISO ..... International Organization for Standardization
JMF ..... Java Media Framework
LDAP ..... Lightweight Directory Access Protocol
LI ..... Length Indicator

MBONE ..... Multicast Backbone
MIME ..... Multipurpose Internet Mail Extensions
MMS ..... Microsoft Media Server Protocol
NAT ..... Network Address Translator
NFS ..... Network File System
NFSNET ..... National Science Foundation Network
Net/3 ..... BSD Networking Software, release 3.0
OSI ..... Open Systems Interconnection
OMA ..... Open Mobile Alliance
PCI ..... Protocol Control Information
PDU ..... Protocol Data Unit
PGI ..... Parameter Group Identifier
PI ..... Parameter Identifier
PV ..... Parameter Value
QoS ..... Quality of Service
RARP ..... Reverse Ardes Resolution Protocol
RDSI ..... Rede Digital de Serviços Integrados
RFC ..... Request for Comments
RM-OSI ..... OSI Reference Model
RTCP ..... Real-Time Transport Control Protocol
RTP ..... Real-Time Transport Protocol
RTSL ..... Real-Time Systems Laboratory
RTSP ..... Real-Time Streaming Protocol
SAP ..... Service Access Point
SBA ..... Subconjunto Básico de Atividade
SBC ..... Subconjunto Básico Combinado
SBS ..... Subconjunto Básico Sincronizado
SDP ..... Session Description Protocol
SDR ..... Session Directory Tool
SESP ..... Session Protocol
SI ..... Session Identifier
SIP ..... Session Initiation Protocol
SMIL ..... Synchronized Multimedia Integration Language
SMTP ..... Simple Mail Transfer Protocol

SPDU ..... Session Protocol Data Unit
SS ..... Session Service
SSL ..... Security Socket Layer
STDs ..... Standards Documents
TCP ..... Transmission Control Protocol
TLS ..... Transport Layer Security
UDP ..... User Datagram Protocol
UML ..... Unified Modeling Language
URL ..... Uniform Resource Locators
VoD ..... Video On Demand
VPN ..... Virtual Private Network
XNS ..... Xerox Network Systems
XTI ..... X/ Open Transport Interface
WAP ..... Wireless Application Protocol
WDP ..... Wireless Datagram Protocol
WSP ..... Wireless Session Protocol
WSP/B ..... Wireless Session Protocol Browsing
WML ..... Wireless Markup Language
WTP ..... Wireless Transaction Protocol
WTLS ..... Wireless Transport Layer Security Protocol
WWW ..... World Wide Web

SUMÁRIO

RESUMO **5**

LISTA DE FIGURAS **7**

LISTA DE TABELAS **8**

LISTA DE QUADROS **9**

ACRÔNIMOS **10**

SUMÁRIO **13**

1 INTRODUÇÃO **16**

1.1 APRESENTAÇÃO 17

1.2 JUSTIFICATIVA 18

1.3 OBJETIVOS 19

1.3.1 Objetivo Geral *19*

1.3.2 Objetivos Específicos *19*

1.3.3 Requisitos *20*

1.4 ORIGINALIDADE, NÃO-TRIVIALIDADE E RELEVÂNCIA DO TEMA 20

1.5 METODOLOGIA DE DESENVOLVIMENTO 21

1.5.1 Pesquisa Bibliográfica *21*

1.5.2 Estudo de Caso *22*

1.5.3 Análise e Interpretação *22*

1.5.4 Desenvolvimento do Modelo Proposto *22*

1.6 LIMITAÇÕES DE ESCOPO 23

1.7 ORGANIZAÇÃO DO TRABALHO 24

2 IDENTIFICAÇÃO DAS FUNCIONALIDADES DE SESSÃO NA ARQUITETURA DA INTERNET **26**

2.1 O CONCEITO DE SESSÃO 26

2.2 A GERÊNCIA DE SESSÃO NO PROTOCOLO HTTP 27

2.3 O CONJUNTO DE PROTOCOLOS SSL/TLS 28

2.3.1 Sessão e SSL *32*

2.4 WIRELESS APLICATION PROTOCOL (WAP) 33

2.4.1 Wireless Session Protocol (WSP) *36*

2.4.2 Modo de Serviço de Sessão com Conexão *38*

2.4.3 Funcionalidades de Sessão Identificadas *40*

2.5 SESSION INITIATION PROTOCOL(SIP) 41

2.5.1 Funções do Protocolo SIP *42*

2.6 ANÁLISE COMPARATIVA 43

2.6.1 RM-OSI versus HTTP *44*

2.6.2 RM-OSI versus SSL *44*

2.6.3 RM-OSI versus WAP *45*

2.6.4 RM-OSI versus SIP *46*
2.6.5 SIP versus WAP e SSL *46*
2.7 CONSIDERAÇÕES FINAIS 47

3 ANÁLISE DAS APLICAÇÕES MULTIMÍDIA 49

3.1 NECESSIDADES DAS APLICAÇÕES MULTIMÍDIA 49
3.2 TRÁFEGO MULTIMÍDIA DA INTERNET 50
3.2.1 Utilização do Tráfego Multimídia *50*
3.2.2 Tráfego Inelástico *52*
3.3 PROTOCOLOS MULTIMÍDIA 53
3.3.1 RTP (Real-Time Transport Protocol) *54*
3.3.2 Real-Time Transport Control Protocol (RTCP) *54*
3.3.3 Real Time Streaming Protocol (RTSP) *55*
3.3.4 Session Description Protocol (SDP) *56*
3.4 UTILIZAÇÃO DOS PROTOCOLOS MULTIMÍDIA 57
3.5 TÉCNICA DE BUFFERIZAÇÃO 59
3.6 SOLUÇÕES PARA APLICAÇÕES MULTIMÍDIA 60
3.7 CONSIDERAÇÕES FINAIS 62

4 A CAMADA DE SESSÃO PROPOSTA 63

4.1 VISÃO GERAL 63
4.2 ELEMENTOS DE COMUNICAÇÃO ENTRE CAMADAS 64
4.2.1 Considerações Sobre o Uso do UDP *66*
4.3 COMPONENTE DE CONTROLE DE SESSÃO (CS) 66
4.3.1 Estabelecimento de Conexão de Sessão *68*
4.3.2 Liberação de Conexão de Sessão *70*
4.3.3 A Recuperação de uma Sessão Existente *71*
4.3.4 Relatório de Anomalias *72*
4.3.5 Controle de Sessão Multiponto *72*
4.4 COMPONENTE FUNÇÕES DE SESSÃO (FS) 74
4.4.1 Transferência de Dados *75*
4.4.2 Sincronização e Ressincronização *77*
4.4.3 Gerência de Diálogo *78*
4.4.4 Gerência de Atividades *80*
4.5 COMPONENTE GERÊNCIA DE BUFFER (GB) 82
4.5.1 A Função de Player Local *83*
4.5.2 A Função de Player Distribuído *85*
4.6 COMPONENTE ANÁLISE DE TRÁFEGO (AT) 88
4.6.1 Funcionamento do Componente AT *89*
4.6.2 Divisão do Tráfego em Elástico e Inelástico *90*
4.6.3 Divisão do Tráfego em Várias Categorias *91*

4.7 INTEROPERABILIDADE 92
4.8 EXTENSÕES FUTURAS 93
4.9 CONSIDERAÇÕES DE IMPLEMENTAÇÃO 94
4.10 CONSIDERAÇÕES FINAIS 95

5 CONCLUSÕES 96

5.1 AVALIAÇÃO DOS RESULTADOS 96
5.2 CONTRIBUIÇÕES DO TRABALHO 98
5.3 TRABALHOS FUTUROS 99

6 REFERÊNCIAS BIBLIOGRÁFICAS 102

APÊNDICE A – ESPECIFICAÇÃO UML 109

A.1 ESTABELECIMENTO DE CONEXÃO DE SESSÃO. 110
A.2 LIBERAÇÃO NEGOCIADA DA SESSÃO 113
A.3 LIBERAÇÃO ABRUPTA DE SESSÃO 116
A.4 ENVIO DE DADOS 119
A.5 FUNÇÃO DE SINCRONIZAÇÃO 122
A.6 SERVIÇO DE CONTROLE DE DIÁLOGO 126
A.7 ESTABELECIMENTO DE CSMU 129
A.8 MECANISMO DE PLAYER LOCAL 133
A.9 MECANISMO DE PLAYER DISTRIBUÍDO 136

APÊNDICE B – MODELO DE REFERÊNCIA RM-OSI 139

B.1 O MODELO DE REFERÊNCIA RM-OSI 139
B.1.1 Primitivas de Serviço OSI 142
B.2 A CAMADA DE SESSÃO 143
B.2.1 Gerência de Diálogo 144
B.2.2 Sincronização 145
B.2.3 Gerência de Atividade 146
B.2.4 Unidades Funcionais da Camada de Sessão 147
B.2.5 Primitivas de Serviço de Sessão 148
B.2.6 Protocolo de Sessão 149
B.3 CONSIDERAÇÕES FINAIS 150

APÊNDICE C – A ESTRUTURA DE COMUNICAÇÃO DA ARQUITETURA INTERNET 151

C.1 PROCESSOS USUÁRIOS 154
C.2 KERNEL – CAMADA DE TRANSPORTE E REDE 154
C.2.1 Socket API 156
C.2.2 Socket 156
C.5 CONSIDERAÇÕES FINAIS 158

ANEXO A – AN EXTENDED MODEL FOR TCP/IP ARCHITECTURE 160

1 INTRODUÇÃO

Atualmente, observa-se a tendência da maioria das empresas em utilizar a Internet como uma ferramenta de suporte à comunicação empresarial. Cada vez mais as empresas dependem de uma infra-estrutura de informática que garanta agilidade no transporte das informações, o que deve ser feito de uma forma segura e eficiente, que sirva de base para a tomada de decisões.

Segundo Gallo e Hancock (2003)

> Toda a história da Internet nos últimos 30 anos evoluiu de uma rede de um projeto de pesquisa do Ministério de Defesa dos EUA (ARPANET) para um meio de comunicação para universidades, governo e pesquisas (NSFNET), para uma rede comercial global, conectando dezenas de milhares de usuários, empresas, escolas e outras organizações. Hoje quase todos os países do mundo estão de alguma forma conectados a Internet. Inicialmente, a ARPANET era as várias redes ancoradas na NFSNET. Hoje, tanto a ARPANET quanto a NFSNET têm aberto espaço para a Internet comercial, que está transformando a Internet de conjunto de redes para uma rede global.

A Internet possui uma infinidade de atrativos, entre os quais podem ser destacados os serviços de correio eletrônico, as transferências de arquivos, as ferramentas de busca e os recursos de comunicação, tais como bate-papo, notícias *online* e jogos interativos. Também existem empresas que desenvolvem em seus *sites* a prática de comércio eletrônico e utilizam a rede como veículo de vendas. Com tantas funcionalidades, o departamento de informática já é uma realidade em algumas empresas e atende tanto os usuários internos quanto fornecedores e clientes.

Hoje, utilizar a Internet é uma necessidade, pois, em muitos casos, a agilidade na execução de atividades empresariais depende dessa estrutura. Entretanto, a estrutura inicial da Internet, projetada há mais de vinte anos, não previa suporte para a demanda tecnológica atual. Por isso, adaptações vêm sendo feitas constantemente, a fim de atender aos requisitos das atuais aplicações.

Um dos objetivos da Engenharia de Produção é o desenvolvimento de métodos de trabalho mais eficazes, que tornem os processos de uma organização empresarial mais eficientes, em busca de uma melhoria da qualidade e produtividade da empresa (CARDOSO, 1995). Neste contexto, esta tese busca uma melhoria na Internet, identificando problemas em sua infra-estrutura e propondo as alterações necessárias para melhorar a sua utilização no âmbito empresarial.

1.1 Apresentação

Desde a concepção da Internet, foram muitas as evoluções e alterações realizadas nas aplicações. Em comparação, pouco foi feito em sua estrutura-base (camada de transporte e camada de rede) que pudesse auxiliar no desenvolvimento e suporte das novas aplicações. Entre as propostas e alterações significativas pode-se referenciar o *Network Address Translator* (NAT), o *Internet Protocol* versão 6 (IPv6) e, mais recentemente, têm-se as propostas de atualizações do protocolo TCP conhecidas como TCP Like e Fast TCP. O NAT e o IPv6 foram alternativas adotadas para solucionar o problema de insuficiência de endereços IP, decorrente do crescimento massivo da Internet, da má distribuição de endereços IP e do próprio modelo de endereçamento IP. O TCP Like e o Fast TCP são alterações realizadas no algoritmo de transmissão de dados do protocolo TCP que buscam adaptar as características de transmissão de dados às condições de tráfego, onde o objetivo é transmitir o máximo de dados possível num curto espaço de tempo.

Observa-se que os protocolos de transporte e de rede da Internet são cuidadosamente projetados, o que não ocorre com os protocolos da camada de aplicação, cujo desenvolvimento visa atender necessidades específicas das aplicações. Além disso, a camada de aplicação da Internet agrega funcionalidades das últimas três camadas do modelo RM-OSI (sessão, apresentação e aplicação). Como resultado, tem-se uma camada de aplicação "inflada", onde funcionalidades idênticas são reimplementadas em vários protocolos e aplicações.

Segundo Tanenbaum (2003),

> A segunda razão para que o OSI não vingasse estava nas falhas do modelo e dos protocolos. A escolha de sete camadas foi mais política do que técnica. Duas camadas (a de sessão e a de apresentação) **estão praticamente vazias**, enquanto duas outras (de enlace de dados e de rede) se encontram sobrecarregadas.
>
> [...] Em resumo, apesar de seus problemas, o modelo OSI (sem as camadas de sessão e apresentação) mostrou-se excepcionalmente útil para a discussão das redes de computadores [...].

Autores de renome discordam da existência das camadas de sessão e apresentação, utilizando um modelo híbrido composto de cinco camadas (aplicação, transporte, rede, enlace de dados e física) como base para o estudo da área de redes de computadores. Porém, acredita-se que a academia deve ter como um de seus principais objetivos questionar e reverter o conhecimento gerado por ela, em contribuição à solução de problemas.

Este trabalho apresenta fatos que comprovam que, embora não exista formalmente uma camada de sessão na arquitetura TCP/IP, suas funcionalidades estão presentes e são essenciais

para o bom funcionamento da Internet. Nesse sentido, este trabalho propõe uma extensão à atual arquitetura da Internet, agregando as funcionalidades de controle de sessão em uma camada separada, capaz de auxiliar o desenvolvimento das aplicações futuras, melhorando a usabilidade e a portabilidade, sem necessidade de alterar as implementações existentes.

1.2 Justificativa

A definição do tema deste trabalho foi realizada em função de algumas razões. Primeiro, acredita-se que seja possível obter vantagens organizacionais e funcionais ao se separarem os serviços de controle de sessão em uma camada específica.

O problema estudado neste trabalho busca atender à demanda de aplicações futuras, pois se considera que serviços de controle de sessão presentes em aplicações já consolidadas, tais como o *Network File System* (NFS), Samba[1], *Security Socket Layer* (SSL) e outras, não precisam ter seu funcionamento adaptado.

Entre as aplicações futuras que podem usufruir os serviços de uma camada de sessão, destacam-se as aplicações multimídia. De acordo com Sharda (1999), esforços serão focados na melhoria das tecnologias de rede existentes, buscando fornecer um serviço melhor na transmissão de dados multimídia. Esses esforços serão baseados em três abordagens. A primeira focará o projeto da infra-estrutura de rede. A segunda se preocupará com os protocolos de rede, e a terceira focará a redução do tráfego, mediante a utilização de técnicas de compactação. Esse trabalho está contextualizado na segunda abordagem, avaliando as alternativas realizadas na construção e adaptação dos protocolos existentes e compondo uma extensão que separa as funcionalidades de controle de sessão em uma camada composta de um ou mais protocolos.

Acredita-se que separando tais funcionalidades seja possível desenvolver técnicas de controle de sessão adaptadas para cada tipo de aplicação. Considera-se como base para a identificação e seleção das funcionalidades de sessão os serviços de sessão definidos na camada de sessão do modelo de referência RM-OSI. Dessa forma, quaisquer problemas relacionados à sessão poderão ser tratados de uma forma isolada, facilitando o desenvolvimento das aplicações na medida em que estas não precisarão mais agregar mecanismos com essa finalidade em seu código-fonte.

---

[1] Samba é um conjunto de programas que implementam o protocolo *Server Message Block* (SMB) nos sistemas Unix.

Acredita-se que essa extensão à arquitetura da Internet contribuirá na solução de problemas que virão auxiliar no atendimento da demanda tecnológica não prevista quando da criação da Internet, e que esta extensão possa impulsionar estudos futuros em que o mesmo possa ser feito em relação à camada de apresentação. Segurança e criptografia são conceitos cada vez mais presentes no desenvolvimento de aplicações Web, onde atualmente a pilha de protocolos SSL é muito utilizada. Entretanto, para o tráfego de vídeo essa alternativa é ineficiente, devido ao alto processamento requerido pelos algoritmos de criptografia. Nestes casos, uma camada de apresentação pode ser a solução.

1.3 Objetivos

1.3.1 Objetivo Geral

Estender a arquitetura da Internet de forma a compor um ambiente que, a exemplo do modelo RM-OSI, agregue as funcionalidades de uma camada de sessão que, no contexto atual da Internet, estão embutidas nas aplicações.

1.3.2 Objetivos Específicos

Para o alcance do objetivo geral, consideraram-se os seguintes objetivos específicos:

a) identificar as funcionalidades de sessão definidas no modelo RM-OSI;
b) identificar as funcionalidades de sessão utilizadas pelas aplicações e protocolos da Internet;
c) verificar a estrutura e funcionamento do kernel do sistema operacional Unix, os mecanismos de sockets utilizados na troca de informações entre sistemas co-operantes e as bibliotecas de sistemas definidas no BSD-Unix. A finalidade é obter o conhecimento necessário em termos de implementação destes sistemas que permita inserir uma camada de sessão na arquitetura da Internet;
d) avaliar as implementações das aplicações e protocolos multimídia no que corresponde aos serviços de sessão;
e) identificar os problemas que podem ser solucionados, as dificuldades, as vantagens e desvantagens que se pode ter com a inserção de uma camada de sessão no contexto atual da Internet; e

f) especificar uma extensão à arquitetura da Internet que contemple uma camada de sessão genérica e incremental. A extensão proposta será modelada na linguagem *Unified Modeling Language* (UML).

1.3.3 Requisitos

Como forma de facilitar a aceitação da arquitetura estendida, estabeleceu-se que o trabalho resultante atenda aos seguintes requisitos:

a) que a arquitetura estendida possa coexistir com a arquitetura atual;
b) que seja portável e leve, permitindo sua rápida disseminação; e
c) que contribua no desenvolvimento de novas aplicações, evitando redundância de códigos.

1.4 Originalidade, Não-Trivialidade e Relevância do Tema

Não se encontrou na literatura, com tudo que se pôde apurar, propostas ou alternativas que estivessem relacionadas a dois pontos: primeiro, na comprovação da existência dos serviços de sessão na Internet; e, segundo, na definição de uma extensão à atual pilha da Internet que contemple os serviços de sessão de forma a auxiliar no desenvolvimento de futuras aplicações.

O ambiente de redes de computadores é bastante complexo. A não-trivialidade deste trabalho surge no momento em que se faz necessário entender o funcionamento da rede e das aplicações de rede que se comunicam. De posse desse conhecimento, pode-se propor alterações que venham a contribuir para melhorar a infra-estrutura existente.

Acredita-se que reestruturar a pilha da Internet, separando serviços que são utilizados por todas as aplicações em uma camada específica, comporá uma base simplificada para o desenvolvimento de novas aplicações.

Propor a inserção de uma camada de sessão na Internet é um desafio que se inicia nesta tese. Esta proposta constitui uma solução inicial que deverá ser incrementada por contribuições da academia e demais organismos de pesquisa nacionais e internacionais.

1.5 Metodologia de Desenvolvimento

"Para que o conhecimento possa ser considerado científico, torna-se necessário identificar as operações mentais e técnicas que possibilitaram a sua verificação" (GIL, 1999, p. 26). Este capítulo apresenta o método utilizado na presente pesquisa de tese.

Todas as ciências caracterizam-se pela utilização de métodos (LAKATOS; MARCONI, 1985). Os métodos que proporcionam as bases lógicas da investigação esclarecem acerca dos procedimentos lógicos que deverão ser seguidos no processo de investigação científica dos fatos. Para o alcance dos objetivos propostos nesta pesquisa utilizou-se o método indutivo, que parte da análise do particular e coloca a generalização como produto posterior do trabalho de coleta de dados particulares. No raciocínio indutivo a generalização deve ser constatada a partir da observação de casos concretos suficientemente confirmadores dessa realidade (GIL, 1999). Dessa forma, através da análise das aplicações e protocolos da Internet, definiram-se as premissas para auxiliar na solução do problema desta tese: quais são os recursos de sessão utilizados pelas aplicações e protocolos que fazem parte da arquitetura da Internet?

Com o objetivo de proporcionar ao investigador meios técnicos para garantir a objetividade e a precisão no estudo dos fatos e oferecer a orientação necessária referente à obtenção, processamento e validação dos dados pertinentes à problemática que está sendo investigada, optou-se por utilizar o método comparativo. O método comparativo visa ressaltar as diferenças e similaridades entre os objetos investigados (GIL, 1999). Dessa forma, assim como o método investigativo, o método comparativo também é utilizado no desenvolvimento deste trabalho, tendo como objetos investigados as arquiteturas de rede existentes e os serviços e funcionalidades de sessão contemplados por elas.

Quanto à finalidade da pesquisa, a pesquisa científica desenvolvida nesta tese é caracterizada como aplicada, por ter como interesse a aplicação e utilização dos resultados. Dentro dos grupos de pesquisa existentes têm-se os estudos exploratórios, estudos descritivos e estudos que verificam hipóteses causais (SELLTIZ at., 1962). Nesta tese utiliza-se o nível de pesquisa exploratória, que tem como finalidade principal desenvolver, esclarecer e modificar conceitos e idéias, envolvendo levantamento bibliográfico e estudos de caso.

1.5.1 Pesquisa Bibliográfica

A pesquisa bibliográfica é desenvolvida a partir de material já elaborado, constituído de livros, artigos científicos, normas de padronização, teses e dissertações. Parte dos estudos exploratórios pode ser definida como pesquisas bibliográficas, assim como certo número de

pesquisas desenvolvidas a partir da técnica de análise de conteúdo (GIL, 1999). A pesquisa bibliográfica permitiu gerar uma base teórica sólida sobre o conceito de sessão, essencial para formular o problema, especificar os objetivos e constituir o tema deste trabalho.

1.5.2 Estudo de Caso

O estudo de caso é um estudo empírico que investiga um fenômeno atual dentro de seu contexto de realidade e pode ser utilizado em pesquisas exploratórias (GIL, 1999).

Este trabalho tem como objeto de estudo de caso o comportamento de algumas aplicações, protocolos e arquiteturas da Internet cujo limite de avaliação é restrito às funcionalidades de sessão. As informações coletadas nos estudos de caso são discutidas e interpretadas pela autora, que realiza juntamente com a pesquisa bibliográfica uma análise comparativa entre os casos avaliados e os conceitos de sessão.

1.5.3 Análise e Interpretação

O conteúdo do código foi analisado em três fases: pré-análise; exploração do material; e tratamento dos dados, inferência e interpretação (BARDIN, 1977, p95).

A pré-análise é a fase da organização. Inicia-se, geralmente, com os primeiros contatos com o código das implementações. A seguir, procede-se com a escolha dos estudos de caso a serem utilizados, a formulação de hipóteses e a preparação do código a ser analisado.

A fase de exploração constitui na identificação e classificação das funcionalidades de sessão encontradas dentro das aplicações e protocolos selecionados.

O tratamento dos dados, a inferência e a interpretação, por fim, objetivam tornar os dados válidos e significativos (GIL, 1999). Para tanto são utilizados quadros comparativos que sintetizam e destacam as informações obtidas. Na medida em que as informações são comparadas pode-se chegar a generalizações, tornando a análise do conteúdo do código um fator importante para o desenvolvimento do modelo.

1.5.4 Desenvolvimento do Modelo Proposto

A pesquisa bibliográfica, somada aos resultados da análise e interpretação dos estudos de caso selecionados, fornece subsídios para o desenvolvimento do modelo proposto nesta tese. Também se considera o conhecimento empírico da autora nesta tese, que nas suas

atividades profissionais e acadêmicas adquiriu fundamentação teórica e prática na área de redes de computadores e multimídia. As atividades profissionais consistem na atuação como administradora de redes, área em que nos últimos dez anos, adquiriu conhecimento em sistemas operacionais Unix e redes padrão TCP/IP. Entre as atividades acadêmicas pode-se citar a dissertação de mestrado (BORTOLUZZI, 1999) e artigos publicados em congresso nacional e internacional (BORTOLUZZI, 2000; BORTOLUZZI, 2004), cujos temas abrangem as duas grandes áreas.

Após a implementação da primeira versão da extensão proposta, que resultou num trabalho de conclusão de curso de graduação em Ciências da Computação (HENCHEN, 2000), optou-se pela utilização da *Unified Modeling Language* (UML).

> A UML é uma linguagem gráfica utilizada para visualização, especificação, construção e documentação de artefatos de sistemas complexos de software. (BOOCH; RUMBAUGH; JACOBSON, 2000).

Sabe-se que nenhum sistema de software de rede interage sozinho. Todo sistema tem atores ou autômatos que esperam que o sistema se comporte de acordo com a maneira prevista. Neste sentido, a UML utiliza casos de uso para especificar o comportamento do sistema ou parte de um sistema descrevendo o conjunto de seqüências de ações realizadas pelo sistema para produzir um resultado observável pelo ator. Casos de uso são usados para denotar somente o comportamento essencial do sistema ou subsistema, sem ser necessário especificar como esse comportamento é implementado (BOOCH; RUMAUGH; JACOBSON, 2000).

Após a visualização e descrição do funcionamento da extensão proposta nesta tese, casos de uso são utilizados para denotar o comportamento essencial do sistema.

1.6 Limitações de Escopo

Este trabalho propõe uma extensão da arquitetura da Internet envolvendo a inclusão da camada de sessão. Considerando o funcionamento das aplicações existentes há mais de 20 anos, não se pretende criar uma nova arquitetura e muito menos levantar questões tomando partido sobre qual dos modelos é melhor: RM-OSI ou TCP/IP. O foco está direcionado à proposta genérica de uma camada de sessão na arquitetura vigente adotada na Internet que possa solucionar problemas atuais não triviais de uma forma organizada.

Problemas de segurança relacionados à integridade e segurança dos dados, isto é, se o dado enviado foi efetivamente o dado recebido, não são abordados no escopo deste trabalho. A apresentação dos dados, como o nome mesmo diz, está ligada à camada de apresentação, saindo do escopo da proposta.

1.7 Organização do Trabalho

Este trabalho está organizado em quatro capítulos, três apêndices e um anexo, iniciando-se com o capítulo corrente, que apresenta o problema a ser solucionado, as justificativas, a motivação, os objetivos propostos nesta tese e a metodologia de pesquisa utilizada. Os demais capítulos, apêndices e anexo apresentam os conteúdos temáticos descritos a seguir.

O Capítulo 2 apresenta o conceito de sessão, indicando como e onde essas funcionalidades são encontradas na Internet. É realizada uma comparação com o modelo de referencia RM-OSI, permitindo identificar a presença dos serviços de sessão na arquitetura TCP/IP. Tal identificação contrapõe a argumentação de autores que definem a camada de sessão como uma camada vazia.

O Capítulo 3 faz um levantamento sobre as necessidades e as redundâncias de funções encontradas nas aplicações multimídia. Descreve o funcionamento dos principais protocolos multimídia, identificando os serviços de sessão presentes em suas funcionalidades. Um quadro comparativo apresenta as redundâncias de código presente nos estudos de casos selecionados. Também são abordados outros assuntos atuais ligados à transmissão de dados multimídia, tais como a dificuldade de identificação do tráfego multimídia na Internet e as propostas de alterações que vêm sendo feitas nos protocolos de transporte da rede, a fim de adequá-los à nova demanda de aplicações e às novas tecnologias físicas de transmissão de dados utilizadas na Internet 2.

O Capítulo 4 descreve a extensão proposta à arquitetura da Internet, que contempla serviços de sessão para a camada de aplicação e fornece subsídios que permitem identificar e categorizar o tráfego da rede.

O Capítulo 5 discute os resultados alcançados, destaca as contribuições deste trabalho sob o ponto de vista inovador e de fomento de pesquisa, e propõe um conjunto de trabalhos futuros que poderão ser desenvolvidos para dar continuidade à solução apresentada.

Para direcionar o foco do trabalho, parte da pesquisa bibliográfica e os casos de uso que permitem identificar o funcionamento do sistema da camada de sessão proposta foram transferidos para os apêndices. Comparações com o RM-OSI e referências sobre o

funcionamento da arquitetura TCP/IP são feitas nos Capítulos 2, 3, 4 e 5 deste trabalho. Sempre que houver necessidade de esclarecimento, o leitor pode consultar os apêndices, que apresentam os conteúdos temáticos descritos a seguir.

O Apêndice A demonstra aplicações da extensão proposta através da construção de casos de uso, que na linguagem UML é a ferramenta utilizada para modelar o comportamento do sistema. Os casos de uso servem para auxiliar na validação da extensão proposta.

O Apêndice B realiza uma revisão bibliográfica sobre o modelo de referência RM-OSI, composto de sete camadas. A quinta camada define os serviços de sessão. Nesta revisão bibliográfica descreve-se o funcionamento do modelo RM-OSI de uma forma genérica e mais profundamente o funcionamento da camada de sessão.

O Apêndice C apresenta a estrutura de comunicação da arquitetura Internet no sistema operacional BSD Unix, onde foi feita a primeira implementação da pilha TCP/IP. Para propor uma extensão à arquitetura da Internet é necessário conhecer como as camadas são implementadas no sistema operacional. Esta visão prática facilita a adequação do modelo à realidade.

O Anexo A apresenta artigo publicado em congresso internacional (*Ninth IEEE Singapore International Conference on Communication Systems*), realizado de 6 a 8 de setembro de 2004, onde se propôs uma extensão à arquitetura da Internet, a partir da inserção de uma camada de sessão.

2 IDENTIFICAÇÃO DAS FUNCIONALIDADES DE SESSÃO NA ARQUITETURA DA INTERNET

Este capítulo descreve as funcionalidades de sessão encontradas em algumas das aplicações, padrões e protocolos da Internet. O objetivo principal é mostrar que sessão existe e está presente na maioria das aplicações atuais. Identificando essas funcionalidades, justifica-se a proposta de inclusão de uma camada de sessão na arquitetura TCP/IP.

O capítulo inicia-se com uma breve descrição do que é sessão, destacando a diferença existente entre sessão e conexão. Mais detalhes sobre a definição de sessão estão descritos no Apêndice B.

As principais funcionalidades de sessão descritas neste capítulo são:

a) gerenciamento de sessão ou armazenamento de estado, que permite o controle da conexão estabelecida entre uma aplicação remota e um servidor;
b) tratamento de mudança de área (*roaming*), que garante a uma estação móvel uma sessão persistente; e
c) inicialização de sessão, que permite que um usuário, contendo seu identificador de sessão, realize chamadas telefônicas sobre uma rede IP, utilizando também serviços de transmissão de dados, voz e vídeo.

Essas funcionalidades serão detalhadas nas seções 2.2 a 2.5.

O capítulo termina com uma análise comparativa em que os serviços de sessão encontrados na Internet são comparados aos serviços de sessão definidos pelo modelo de referência RM-OSI. Uma segunda comparação também é feita entre os estudos de casos selecionados.

2.1 O Conceito de Sessão

Como descrito no modelo de referência RM-OSI (Apêndice B), o serviço de sessão do modelo RM-OSI deve prover os meios necessários para a troca organizada e sincronizada de dados entre duas entidades pares cooperantes.

Para facilitar o entendimento, faz-se necessário enfatizar a diferença existente entre conexão de transporte e conexão de sessão (ou simplesmente sessão). Conexão de transporte é um canal de comunicação específico, mapeado sobre o TCP ou UDP, que no sistema

operacional Unix é implementado com sockets cujo funcionamento é descrito no Apêndice C. Sessão é uma construção virtual existente entre um cliente e um servidor. Múltiplas conexões podem ser associadas a uma dada sessão e múltiplas sessões podem ser associadas a uma conexão. A Figura 1 demonstra esse relacionamento.

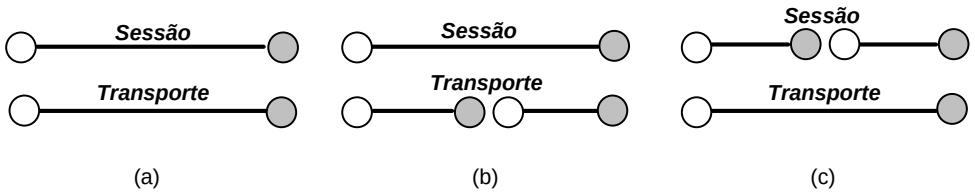


Figura 1 – Diferentes relações entre conexão de sessão e de transporte: (a) correspondência um a um; (b) várias conexões de transporte para uma única sessão; (c) uma conexão de transporte para várias sessões. Fonte: Adaptado de Tanenbaum (1991).

Devido à ausência de uma camada de sessão na arquitetura da Internet, aplicações implementam, dentro de seu código, mecanismos de estabelecimento, gerenciamento e término de sessões lógicas, que buscam atender especificamente a cada uma de suas necessidades. Para cada sistema um mecanismo específico é desenvolvido, gerando problema de reimplementação e redundância de código.

2.2 A Gerência de Sessão no Protocolo HTTP

O protocolo HTTP, por exemplo, é um protocolo simples, baseado em pedido e resposta, que não possui armazenamento de estado (*stateless*) (KRISTO; MONTULLI, 1997). Aplicações baseadas em HTTP que necessitem de controle de sessão normalmente utilizam cabeçalhos *cookies*, parâmetros escondidos de formulários ou reescrita *Uniform Resource Locator*[2] (URLs) (KURNIAWAN, 2002). Também podem ser utilizadas abstrações de mais alto nível, tais como as implementadas em *frameworks* para Web, como, por exemplo, objetos HttpSession em *Java Servlets* e *Python Session Objects* na extensão Python para Apache (estes dois usam *cookies* para identificação de sessões).

A Figura 2 e o Quadro 1 demonstram um exemplo de uma aplicação Web que faz o seu controle de sessão utilizando o mecanismo de *cookies*.

[2] *Uniform Resource Locator* (URLs) é o endereço global de documentos e outros recursos no *World Wide Web* (WWW).

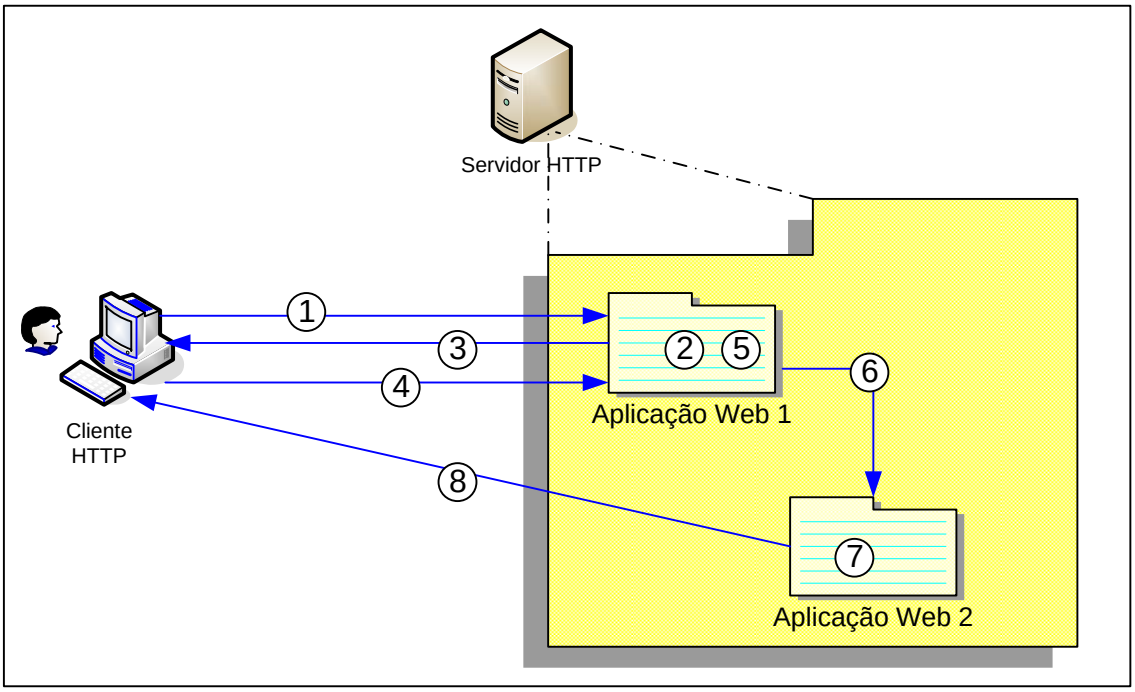


Figura 2 – Visualização dos passos de execução do HttpSession

PASSO	AÇÃO
1	Cliente se autentica em uma aplicação Web.
2	Aplicação Web cria um objeto de sessão.
3	Um cabeçalho *cookie* contendo o identificador de sessão é enviado com a resposta ao cliente.
4,5	Cliente solicita mais um recurso à aplicação Web. Um cabeçalho *cookie* contendo o id_session é enviado com a solicitação.
6,7	A aplicação Web recebe a solicitação e extrai o id_session contido no cabeçalho *cookie*. O objeto sessão é recuperado a partir do id_session.
8	A resposta do recurso solicitado é enviada ao cliente.

Quadro 1 – Descrição da criação e andamento de uma sessão em uma aplicação Web

Se a arquitetura da Internet possuísse uma camada específica que tratasse do gerenciamento de sessões, as aplicações poderiam acessar tais funcionalidades em forma de elementos de serviço, como foi sugerido no modelo de referência RM-OSI.

O próximo item descreve as funcionalidades de sessão encontradas na implementação do conjunto de protocolos *Security Socket Layer* (SSL) / *Transport Layer Security* (TLS).

2.3 O Conjunto de Protocolos SSL/TLS

O protocolo SSL foi desenvolvido inicialmente pela *Netscape Communications*, com o objetivo de colocar uma camada intermediária entre a camada de transporte e a camada de aplicação, para prover serviços de comunicação seguros, conforme ilustrado na Figura 3. O

SSL necessita que a camada de transporte seja robusta, que as mensagens sejam entregues sem erros e na ordem correta, funcionando, assim, sobre o TCP. O SSL foi projetado para proteger qualquer protocolo de aplicação baseado em TCP, tais como os protocolos HTTP, FTP, POP, IMAP4, entre outros (THOMAS, 2000).

O SSL faz distinção entre sessão e conexão. Uma conexão representa um canal de comunicação específico (tipicamente mapeado dentro de uma conexão TCP), junto com suas chaves, escolha de cifras, estado do número de seqüência, etc. Uma sessão é uma construção virtual que representa os algoritmos e parâmetros negociados. Várias conexões podem ser associadas a uma dada sessão, tendo cada conexão sua própria chave de encriptação. Isso é necessário por razões de segurança, já que reutilizar a mesma chave simétrica pode ser extremamente perigoso (RESCORLA, 2001).

O protocolo SSL é formado por cinco outros protocolos, cuja estrutura pode ser visualizada na Figura 3.

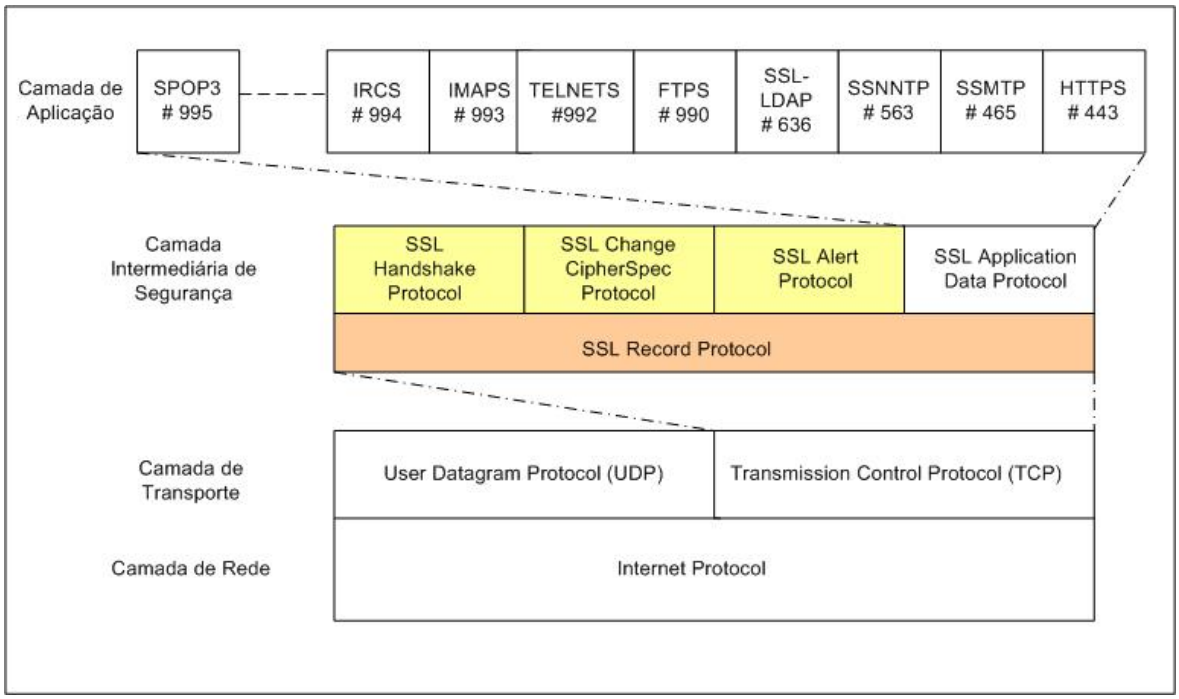


Figura 3 – Composição do protocolo SSL. Fonte: Adaptado de Thomas (2000)

O protocolo *ChangeCipherSpec* possui somente uma mensagem e serve para ativar o processo de encriptação.

O protocolo de alerta (*Alert Protocol*) é responsável por sinalizar erros ou condições de aviso. Existem dois níveis de severidade para esses eventos: nível 1, que indica que a sessão

não deve ser retomada no futuro; e nível 2, que indica que a sessão deve ser interrompida imediatamente.

O protocolo *Handshake* é o principal responsável pela negociação de parâmetros de sessão, tais como chave de encriptação, identidade do servidor, identidade do cliente e outros. Dessa forma, o SSL provê encriptação e autenticidade das mensagens, o que garante confidencialidade e integridade.

O protocolo de dados de aplicação (*Application Data Protocol*) envia as mensagens de dados da aplicação utilizando um canal seguro previamente estabelecido.

Esses quatro protocolos são construídos sobre o protocolo da camada de registros (*Record Layer Protocol*), visualizado na Figura 3, que tem a função de encapsular todas as mensagens SSL, realizando a verificação da integridade das mensagens, encriptação e compactação.

Numa sessão SSL existe um cliente e um servidor, e a sessão sempre é iniciada pelo cliente, que sugere parâmetros de comunicação. O servidor escolhe os parâmetros de comunicação entre aqueles sugeridos pelo cliente. O processo de autenticação é feito pelo cliente; algumas vezes pelo servidor (THOMAS, 2000).

Servidores e clientes usam informações diferentes para estabelecer sessões. Quando um cliente inicializa uma conexão SSL (ClientHello), as únicas informações disponíveis são a informação do nome do host (ou endereço IP, se o nome não for usado) e a porta. O campo identificador de sessão existe, mas é insignificante nessa fase. Quando o servidor envia a resposta (ServerHello), o campo identificador de sessão recebe um valor único, que identifica uma comunicação SSL particular; em outras palavras, uma sessão.

A mais básica função do SSL é o estabelecimento de conexão encriptada. A Figura 4 e o Quadro 2 descrevem a seqüência de passos utilizada no processo de *Handshake*. Mais detalhes podem ser obtidos em Rescorla (2001).

O processo descrito na Figura 4 não testa a autenticidade do servidor. Isso pode ser um problema, pois um site malicioso pode personificar outro site. Para solucionar isso, o servidor deve poder fornecer seu certificado, como forma de verificar sua identidade, e, nesse caso, o campo *ServerKeyExchange* é substituído por *Certificate* (THOMAS, 2000).

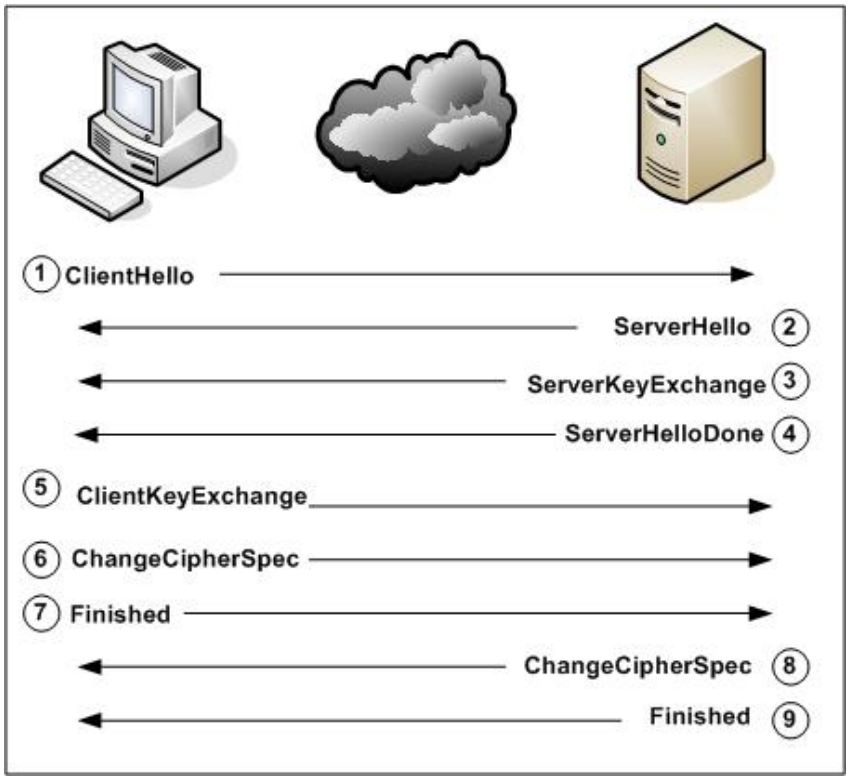


Figura 4 – Estabelecimento de conexão encriptada no SSL. Fonte: Adaptado de Thomas (2000)

PASSO	AÇÃO
1	Cliente envia ***ClientHello*** propondo as opções SSL. ClientHello negocia versão do protocolo, informa o número aleatório para semente em algoritmos de criptografia, a lista de algoritmos de criptografia aceitáveis, os respectivos tamanhos das chaves e os métodos de compactação possíveis. A compactação deve ser feita antes de encriptar.
2	Servidor responde com ***ServerHello*** selecionando as opções SSL. ServerHello estipula a versão utilizada, informa número aleatório, ID de sessão, algoritmo a ser usado, tamanho de chave e os métodos de compactação.
3	Servidor envia sua chave pública em ***ServerKeyExchange*.** A chave pública do servidor depende do algoritmo de criptografia selecionado em ***ServerHello***.
4	Servidor termina sua negociação inicial em ***ServerHelloDone***.
5	Em ***ClientKeyExchange*** o cliente envia uma chave simétrica para encriptar a sessão. Essa chave é enviada encriptada com a chave pública do servidor.
6	Cliente envia ***ChangeCipherSpec*** para ativar opções negociadas. Essa ação define um algoritmo de criptografia simétrica, um algoritmo de verificação de integridade de mensagem e chaves para algoritmos. Os algoritmos e chaves podem ser diferentes para cada sentido da comunicação.
7	Cliente envia ***Finished*** para que o servidor possa verificar as opções ativadas. Termina o processo de negociação. Essa ação permite verificar a integridade dos parâmetros da sessão negociada.
8	Servidor envia ***ChangeCipherSpec*** para ativar as opções negociadas.
9	Servidor envia ***Finished*** para que o cliente verifique as opções ativadas.

Quadro 2 – Estabelecimento de conexão encriptada no SSL

O restabelecimento de uma sessão SSL exige muita troca de mensagens e processamento, gerando *overhead* (sobrecarga). Esse custo pode ser reduzido se clientes puderem restabelecer sessões enviando um *ClientHello* com o identificador da sessão a ser restabelecida. O servidor indica que concorda com o restabelecimento da sessão se utilizar o mesmo identificador de sessão em *ServerHello*. Esse processo pode ser visualizado na Figura 5.

Uma vez que o identificador de sessão é utilizado no processo de restabelecimento de sessão, não importa mais qual é o endereço IP e a porta que o cliente utilizará. Essa medida foi necessária porque portas TCP são escolhidas aleatoriamente e os endereços IP podem ser alterados em redes com DHCP. Servidores devem suportar isso (RESCORLA, 2001).

Além do estabelecimento de conexão encriptada, outras funções são oferecidas pelo SSL, tais como autenticação do servidor e autenticação do cliente, que permitem teste de identidade.

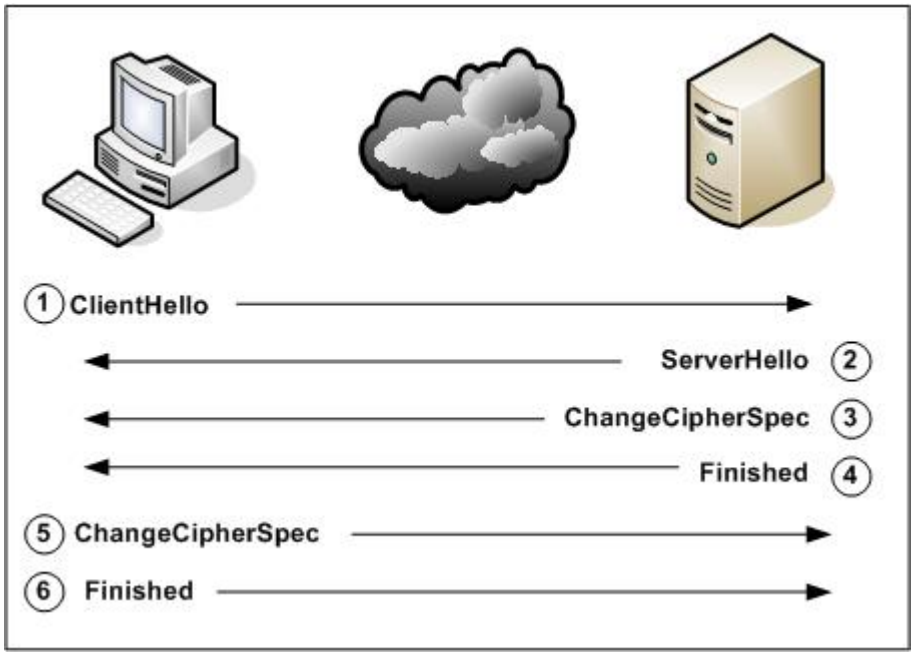


Figura 5 – Restabelecimento de uma sessão no SSL. Fonte: Adaptado de Thomas (2000)

2.3.1 Sessão e SSL

Identificadores de sessão estão presentes tanto nas implementações das aplicações Web (*cookie, servlet*) quanto no protocolo SSL.

O SSL implementa funcionalidades da camada de apresentação e da camada de sessão do modelo RM-OSI. Com relação à camada de apresentação, pode-se citar a compactação dos

dados e o processo de criptografia utilizado, essencial para garantir um canal seguro; e, relacionados à camada de sessão, há a funcionalidade de armazenamento de estado e o relatório de anomalias.

A implementação de armazenamento de estado realizada nas aplicações Web não possui a mesma segurança oferecida pelo SSL, onde se sugere que tal vulnerabilidade seja amenizada utilizando-se outros recursos, entre os quais se pode citar: *Virtual Private Network* (VPN), IP-sec e o próprio SSL, através do protocolo HTTPS.

Visualiza-se na Figura 3 que o SSL funciona como uma camada intermediária entre a aplicação e o transporte, e que implementa recursos de sessão e apresentação. Assim como o SSL, o WAP também implementa recursos de apresentação e sessão, com a ressalva de que no WAP existe mais de uma camada entre a aplicação e o transporte. O conjunto de protocolos WAP será apresentado na seção 2.4, a seguir.

2.4 Wireless Aplication Protocol (WAP)

O protocolo *Wireless Application Protocol* (WAP) é o resultado de um trabalho contínuo que define a especificação para o desenvolvimento de aplicações que operam sobre redes sem fio. Seguindo esse escopo, o WAP *Forum* determinou um conjunto de especificações para serem usadas a serviço das aplicações (WAP-HTTP, 2001).

Os objetivos do WAP *Forum* são (WAP-HTTP, 2001):

a) gerar conteúdo para a Internet e avançar nos serviços de dados de telefonia digital celular e outros terminais sem fio;
b) criar a especificação de um protocolo sem fio global que possa trabalhar com diferentes tecnologias de redes sem fio;
c) capacitar a criação de conteúdo e de aplicações dimensionadas para o amplo alcance de redes portáteis e tipos de dispositivos; e
d) adotar e estender os padrões e tecnologias existentes sempre que apropriado.

Assim como o TCP/IP, o WAP foi projetado em camadas, ilustradas na Figura 6.

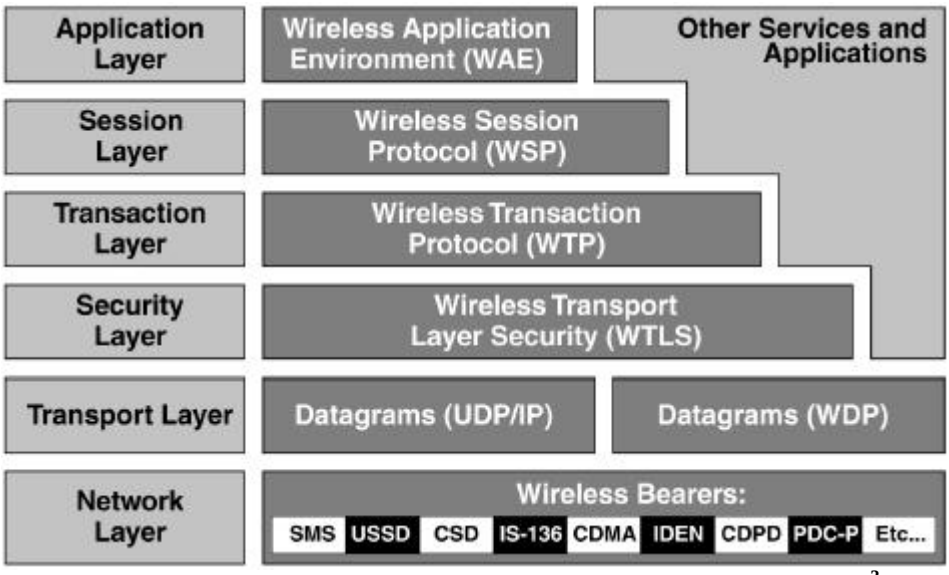


Figura 6 – Arquitetura de protocolos WAP. Fonte: OMA (2003)[3]

Parte da evolução das especificações WAP busca adaptar a pilha WAP aos protocolos definidos pelo IETF, que envolve os seguintes documentos:

a) HTTP 1.1 (FIELDING, 1999);
b) TLS Profile and Tunnelling (WAP-TLS, 2001); e
c) Wireless Profiled-TCP (WAP-TCP, 2001).

A Figura 7 apresenta, com mais detalhes, o funcionamento de uma aplicação WAP. Quando o dispositivo de navegação WAP é utilizado para solicitar informação, o pedido da URL é enviado utilizando-se a pilha de protocolos WAP. Esse pedido é enviado da rede sem fio para um *gateway*, que permite que os usuários da rede *wireless* se conectem à Internet. Ao receber a solicitação, o *gateway* executa algumas tarefas, decodificando e convertendo a solicitação do protocolo WSP para o protocolo da Internet. Essa requisição é enviada a um servidor Web, que recebe e retorna uma resposta contendo conteúdo *Wireless Markup Language* (WML) para o *gateway* WAP. Ao receber o conteúdo VML do servidor Web, o WAP *gateway* codifica, converte e criptografa a informação que será enviada através da rede *wireless* para o navegador WAP. O usuário do dispositivo WAP recebe a informação na tela.

[3] Open Mobile Alliance (OMA) é uma organização independente, que inclui aproximadamente 200 companhias que dominam a área de operadores móveis, dispositivos e suprimentos de rede, tecnologia da informação e conteúdo para provedores de serviço.

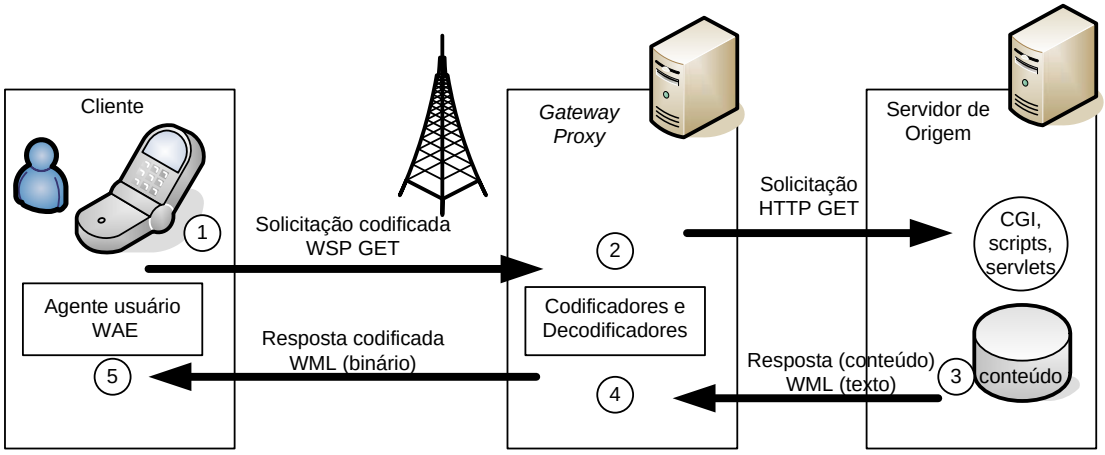


Figura 7 – Funcionamento de uma aplicação WAP. Fonte: Adaptado de Proença (2003)

Este trabalho está contextualizado apenas nas funcionalidades da camada de sessão. Detalhes sobre os demais protocolos da pilha WAP podem ser obtidos em WAP-HTTP (2001).

O modelo em camadas do protocolo WAP é ilustrado na Figura 8 e lembra o modelo do RM-OSI. Entidades de gerenciamento de camada tratam da inicialização do protocolo, da configuração e das condições de erro que não são tratadas pelo próprio protocolo, tais como a perda de conectividade devido à estação móvel estar fora de área.

O protocolo WSP foi projetado para funcionar sobre o serviço de transação e o serviço de datagramas. A segurança deve ser oferecida em uma camada opcional em cima da camada de transporte. Sozinho, o WSP não precisa de segurança, entretanto as aplicações que o utilizam podem vir a precisar, e nesse caso o serviço é oferecido de forma modular, preservando as interfaces do serviço de transporte (WAP-WSP, 2000).

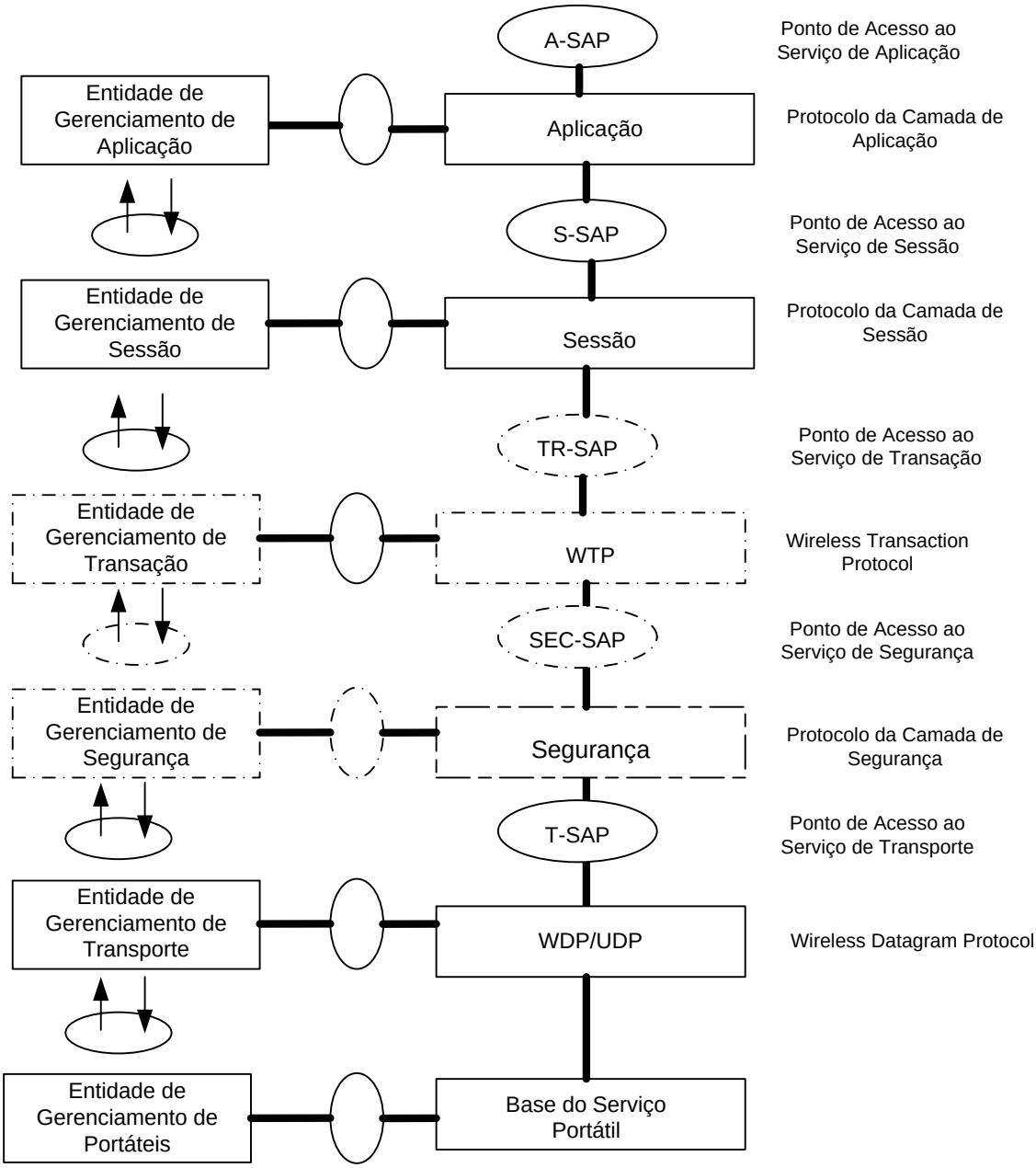


Figura 8 – Modelo de Referência do WAP. Fonte: Adaptado de OMA (2003)

2.4.1 Wireless Session Protocol (WSP)

O *Wireless Session Protocol* (WSP) é a família de protocolos da camada de sessão da arquitetura WAP que executa operações remotas entre um cliente e um servidor *proxy* (*gateway*). O WSP disponibiliza para a camada de aplicação WAP uma interface consistente para dois serviços de sessão. O primeiro é um serviço com conexão que opera sobre o protocolo de transação *Transaction Layer Protocol* (WTP). O segundo é um serviço sem

conexão que opera sobre um serviço de transporte de datagramas seguro ou não seguro (WAP-WSP, 2000).

Relacionado ao contexto de segurança, o WSP não fornece suporte à conexão segura; ele considera que esse suporte é oferecido pelas camadas anteriores (WTLS). As aplicações que precisarem desse suporte devem buscar outra solução, tal como o TLS (WAP-WSP, 2000).

De acordo com a especificação, o WSP possui as seguintes características gerais (WAP-WSP, 2000):

a) estabelece uma sessão confiável entre o cliente e o servidor e libera essa sessão de uma maneira ordenada;
b) utiliza a negociação de capacidade[4] para realizar um acordo sobre as funcionalidades do protocolo que o cliente e o servidor suportam;
c) envia conteúdo entre cliente e servidor usando uma codificação compacta; e
d) suspende e retoma uma sessão.

O foco do WSP é oferecer serviços adequados para aplicações que funcionam em navegadores (*Wireless Session Protocol / Browser* – WSP/B). Ele suporta os métodos e semântica definida no HTTP 1.1, utilizando uma codificação compacta para minimizar o *overhead.* Por exemplo, um dos serviços que o WSP oferece para a camada de aplicação é o serviço de transferência de dados classificados, que funciona como uma extensão, em que o conteúdo do cabeçalho HTTP 1.1 permite identificar o tipo dos dados enviados, o conjunto de caracteres, as linguagens e outros tipos de dados.

O ciclo de vida de uma sessão WSP independe da camada de transporte. Por exemplo, uma sessão pode ser suspensa quando está ociosa, para liberar recursos ou economizar bateria de um dispositivo portátil. A sessão pode ser retomada sem o *overhead* inicial gerado no estabelecimento da sessão, podendo ser restabelecida sobre uma rede portátil diferente.

O WSP oferece dois serviços de sessão: sem conexão e com conexão. O serviço de sessão sem conexão é adequado quando aplicações não precisam de um envio de dados confiável e não estão preocupadas com a confirmação. Esse serviço pode ser usado sem existir realmente uma sessão estabelecida. O provedor de serviço de sessão utiliza diretamente o ponto de acesso ao serviço de transporte (SAP-T), como pode ser visualizado na Figura 8.

[4] "Capacidade" foi traduzido do inglês: *Capability*. Dados de capacidade são dados utilizados na troca de mensagens de controle.

Caso exista necessidade, o provedor de serviço de sessão pode também utilizar o ponto de acesso ao serviço de segurança (SAP-SEC). O serviço de sessão sem conexão não precisa de máquina de estados, já que, nesse modo, cada primitiva do serviço da interface WSP envia os dados diretamente da unidade de dados do protocolo de sessão para a camada de transporte, sem se preocupar com confirmação (WAP-WSP, 2000).

Durante o estabelecimento de uma conexão de sessão, a troca de informação entre o usuário e o fornecedor do serviço de sessão é negociada utilizando-se capacidades. A negociação de capacidades refere-se à facilidade que o protocolo de sessão oferece para a configuração de parâmetros que o cliente e o servidor suportam. A negociação de capacidade, utilizada no serviço com conexão, permite que um servidor de aplicação determine quais facilidades e configurações são suportadas por determinado cliente (WAP-WSP, 2000). O item 2.4.2 detalha as facilidades dos serviços de sessão, no modo orientado à conexão do WSP.

2.4.2 Modo de Serviço de Sessão com Conexão

O serviço de sessão com conexão é dividido em facilidades. Algumas delas são opcionais, e a maioria é assimétrica, porque as operações disponíveis entre o cliente e o servidor, conectados através de uma sessão, são diferentes (WAP-WSP, 2000).

As facilidades oferecidas são (WAP-WSP, 2000):

a) gerenciamento de sessão;
b) invocação de métodos;
c) relatório de exceção;
d) relatório de exceção;
e) facilidade de *push*;
f) facilidade de *push* confirmado; e
g) resumo de sessão.

O gerenciamento de sessão está sempre disponível; ele permite que um cliente conectado a um servidor tenha acesso às facilidades e opções utilizadas pelo protocolo. Um servidor pode reutilizar a tentativa de conexão, redirecionando o cliente para outro servidor. Durante o estabelecimento de sessão, o cliente e o servidor podem enviar atributos de informações, que permanecem válidos enquanto durar a sessão. Ambos, cliente e servidor, podem terminar a sessão; nesse caso, a entidade par é notificada sobre o término da sessão.

Ao ocorrer o término da sessão, o usuário também é notificado, e isso é feito através da ação de um provedor de serviço ou de uma entidade de gerenciamento.

A invocação de métodos permite que um cliente peça para um servidor executar uma operação e retornar o resultado. Isso é eficaz para operações sobre métodos HTTP ou extensões de operações definidas pelo usuário, tais como um padrão de transação. Os usuários do serviço, tanto do cliente quanto do servidor, são sempre notificados sobre a transação completa, tendo ela sucesso ou falha. A falha pode ser causada por uma inicialização que foi abortada, por um usuário de serviço ou um provedor de serviço.

O relatório de exceção está sempre disponível; ele permite que um fornecedor de serviço notifique o usuário sobre eventos que estão relacionados em uma transação particular e não ocasiona uma mudança no estado da sessão.

A facilidade para *push* (empurrão) permite que o servidor envie uma informação não solicitada para o cliente, obtendo vantagem sobre a sessão de informação compartilhada entre o cliente e o servidor. Essa facilidade é a única não confirmada; logo, o envio da informação pode ser não confiável.

A facilidade de *push* confirmado é semelhante à anterior, mas o cliente confirma o recebimento da informação. O cliente pode também escolher abortar o *push*. Nesse caso, o servidor é notificado.

O resumo de sessão inclui meios de suspender a sessão, desde que o estado da sessão seja preservado. Ambas as partes reconhecem que promover a comunicação não será possível, a menos que o cliente refaça o estabelecimento de sessão. Este mecanismo é também usado para tratar de situações nas quais o provedor de serviço detecta que uma comunicação distante ainda pode ser alcançada, desde que o usuário do serviço ou a entidade de gerenciamento tomem algumas ações corretivas. O resumo de sessão pode também ser usado para chavear sessões em redes portáteis alternativas, onde propriedades mais apropriadas devem ser utilizadas. Essa facilidade poderia ser implementada para assegurar comportamento cabível em certos ambientes de redes portáteis.

No modo de serviço de sessão com conexão, o WSP utiliza um identificador de sessão designado pelo servidor para controle de ambos: cliente e servidor. O método utilizado para designar o identificador de sessão deve ser escolhido de forma que não exista um valor repetido durante o tempo de vida de uma mensagem usada na rede de transporte; do contrário, a lógica do gerenciamento de sessão pode ser confundida (WAP-WSP, 2000).

As sessões são associadas entre entidades pares, através da quádrupla (endereço do cliente, porta do cliente, endereço do servidor e porta do servidor). A chegada das transações

está associada a uma sessão particular baseada nessa quádrupla de endereços pares. Conseqüentemente, a quádrupla de endereços pares é o único identificador de sessão em nível de protocolo, e deve existir apenas uma sessão ativa para ela num determinado tempo. O servidor deve garantir isso. Caso exista um pedido para criar uma nova sessão utilizando uma quádrupla de endereços pares existentes, o servidor deve forçar que a instância da sessão em uso processe essa transação (WAP-WSP, 2000).

A seção 2.4.3 destaca funcionalidades de sessão e apresentação identificadas no WSP.

2.4.3 Funcionalidades de Sessão Identificadas

Observa-se que as arquiteturas modernas, tal como o WAP, incluem uma camada de sessão (WSP) e buscam especificá-la seguindo (como base) o modelo RM-OSI.

O modo sem conexão do WSP implementa apenas as facilidades de invocação de métodos e *push* de dados; também não necessita da definição de uma máquina de estados (WAP-WSP, 2000). Partindo do pressuposto de que não existe sessão se não existir uma conexão preestabelecida, o modo sem conexão do WAP não funciona como um serviço da camada de sessão, e sim como uma adaptação que utiliza facilidades do WSP para envio de dados diretamente sobre o UDP, isto é, funciona como um mecanismo de passagem direta.

O protocolo WSP oferece serviços de apresentação tais como a codificação binária dos dados e serviços puramente de sessão que visam garantir a operação de serviços móveis.

Como o protocolo TCP não foi projetado para a atual demanda de transmissão de dados sem fio, houve a necessidade de incluir uma nova camada de transporte (WTP).

O WTP evita que cópias duplicadas de pacotes sejam recebidas por um destinatário e permite retransmissão no caso de perda de pacotes. Nesse sentido, ele é análogo ao TCP; entretanto, o WTP é essencialmente uma simplificação do TCP, que possibilita algum desempenho extra da rede.

Assim como o WTP, a camada de sessão (WSP) também teve que ser inserida com o objetivo de garantir a funcionalidade das aplicações sobre essa nova tecnologia de transmissão de dados em dispositivos portáteis.

2.5 Session Initiation Protocol(SIP)

O protocolo SIP, proposto na RFC 2543 e redefinido na RFC 3261, é um protocolo de sinalização utilizado para estabelecimento, modificação e finalização de sessões em uma rede IP. Grande parte da indústria aceita o SIP como sendo um mecanismo de sinalização padrão para os serviços de chamada de voz e multimídia. Ao contrário do padrão H.323, que é um conjunto completo de protocolos, o SIP é um único módulo que compõe um conjunto de protocolos e serviços necessários para o funcionamento de aplicações multimídia na Internet (ROSENBERG, 2002).

Originalmente, o SIP foi desenvolvido para a sinalização do grande número de conferências multimídia do *Multiparty Multimedia Session Control* (MMUSIC), grupo de trabalho do IETF. Hoje em dia, o principal foco do SIP é em direção a chamadas telefônicas, também referenciadas como VoIP, mas teleconferências também são suportadas pelo SIP (ROSENBERG, 2001).

De acordo com Handley (1999), existem muitas aplicações da Internet que precisam de criação e gerenciamento de sessão. A implementação dessas aplicações é complicada, pois os usuários podem se movimentar entre os pontos finais, podendo ser endereçados por vários nomes e se comunicar em diferentes tipos de mídia (áudio, vídeo e dados) – alguns simultaneamente.

O SIP permite a um participante localizar outro participante e negociar os parâmetros da sessão multimídia. O áudio e o vídeo do conteúdo multimídia são enviados entre os participantes usando-se um protocolo de transporte apropriado. Usualmente utiliza-se o *Real-Time Transport Protocol* (RTP). O SIP trabalha em consenso com todos os protocolos criados para enviar dados de sessões multimídia em tempo real[5], completando uma arquitetura multimídia, com o estabelecimento de uma relação ponto a ponto entre os usuários participantes. Porém, a funcionalidade básica do SIP independe dos protocolos multimídia, pois ele não oferece serviços, e sim primitivas que podem ser utilizadas por diferentes serviços. Ele permite localizar um usuário e enviar um objeto com o conteúdo de sua localização, utilizando a combinação usuário@domínio, uma porta TCP e uma estrutura de servidores *proxy*, na qual os registros dos usuários e outras requisições são armazenados.

[5] Protocolos como RTP, RTSP, MEGACO, SDP e outros.

2.5.1 Funções do Protocolo SIP

O SIP pode ser utilizado em chamadas telefônicas normais, estabelecendo uma conexão ponto a ponto, ou em uma conferência colaborativa, sendo responsável pela execução das seguintes funções: localização do usuário; tradução de nomes; negociação e mudança de atributos de chamada; gerenciamento de participantes de chamada; e composição de protocolos.

A estrutura básica e o funcionamento do protocolo SIP podem ser visualizados na Figura 9, onde os principais componentes da arquitetura são (CISCO SYSTEMS, 2003):

a) agente do usuário SIP;
b) servidor *proxy*;
c) servidor de redirecionamento; e
d) registrador SIP.

O agente do usuário SIP é o terminal, que funciona como cliente no pedido de inicialização de sessão e como servidor ao responder a um pedido de inicialização de sessão. O agente de chamada utiliza a combinação usuário@domínio para realizar chamadas e aceitar chamadas de outro agente do usuário.

O servidor *proxy* funciona como um servidor intermediário, que passa as requisições do agente do usuário SIP para o próximo servidor SIP, e também retém informações com a finalidade de contabilização. O servidor SIP pode dividir as chamadas, por ordem de chegada, para que várias extensões toquem ao mesmo tempo. O primeiro que atender pega a chamada, enquanto as outras extensões param de tocar. Para tentar resolver o pedido de endereço de *host*, o servidor *proxy* SIP pode utilizar métodos de busca DNS, busca em base de dados ou retransmissão para o próximo *host*.

O servidor de redirecionamento SIP tem a função de fornecer a resolução de nome e localização do usuário. O servidor de redirecionamento SIP responde ao pedido do agente usuário, fornecendo informações sobre o endereço do servidor. Dessa forma, o cliente pode estabelecer contato com o servidor, diretamente. Esse funcionamento é semelhante ao redirecionamento de páginas Web.

O registrador SIP fornece um serviço de informação de localidades; ele recebe informações do agente do usuário e armazena essa informação de registro.

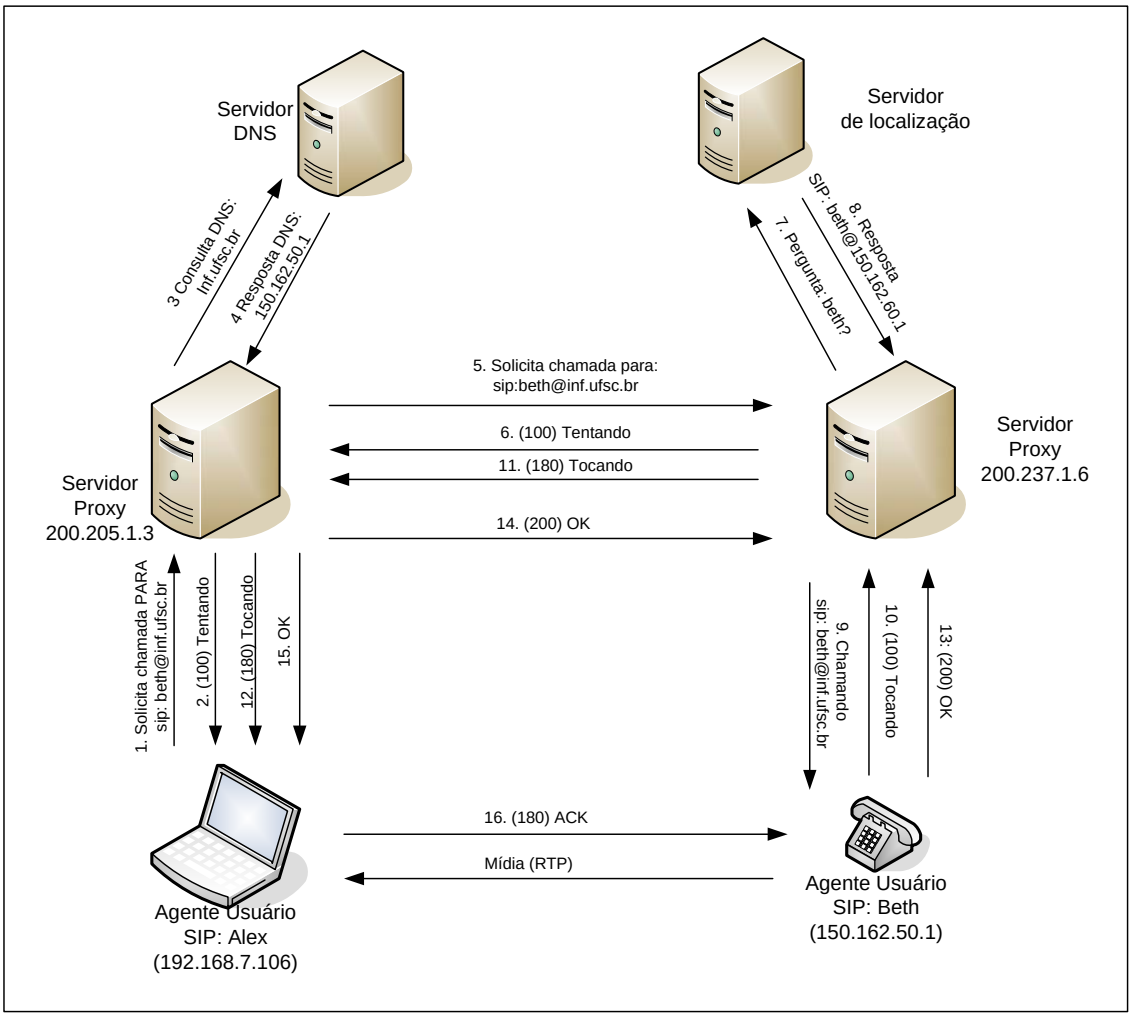


Figura 9 – Funcionamento do SIP. Fonte: Adaptado de Cisco Systems (2003)

Segundo Vide (2005), exemplos de implementação do SIP incluem o “Messenger” da Microsoft, que transforma o PC em um software de telefone (uma voz através de um dispositivo IP) com as ferramentas adicionais de vídeo, *chat* e compartilhamento de dados.

2.6 Análise Comparativa

Buscando enfatizar a presença de sessão na arquitetura da Internet, a seguir é apresentado um comparativo entre cada um dos estudos de caso avaliados e a definição de sessão do modelo de referência RM-OSI, descrita em detalhes no Apêndice B. Também são feitas comparações entre os serviços de sessão presentes nos estudos de caso selecionados.

2.6.1 RM-OSI *versus* HTTP

Aplicações Web necessitam de gerenciamento de sessão (armazenamento de estado) e programam em seu código fonte mecanismos (*cookie e servlet*) para suprir essa necessidade, que foi definida na unidade funcional Kernel da camada de sessão do modelo de referência RM-OSI (descrita no Apêndice B.2.4).

O *cookie*, mecanismo utilizado por grande parte das aplicações Web, contém um campo id_session que é enviado junto com a solicitação do cliente. De modo semelhante, a SPDU (*Session Protocol Data Unit*) do RM-OSI também possui um campo identificador de sessão.

Observam-se nas aplicações de comércio eletrônico e nas aplicações bancárias características de funcionamento que correspondem ao conceito de atividade definido pela quinta camada do RM-OSI. Por exemplo, ao se iniciar uma atividade de transferência bancária, apenas os formulários relacionados a essa atividade estarão disponíveis. Este tipo de aplicação não permite que ao mesmo tempo o usuário (cliente bancário) efetue uma transferência e verifique o saldo disponível em sua conta corrente. Somente na finalização da atividade primeiramente iniciada é possível realizar uma segunda atividade. O conceito de atividades do RM-OSI define que uma sessão pode ter várias atividades consecutivas, porém num dado instante somente uma atividade pode estar em progresso em uma conexão de sessão.

2.6.2 RM-OSI *versus* SSL

Analisando a camada de sessão do modelo RM-OSI e a pilha de protocolo SSL observa-se que o protocolo de alerta (*Alert Protocol*) do SSL possui algumas funcionalidades que podem ser comparadas às dos relatórios de exceção (*Exception Report*) da norma X.225 do CCITT. A norma define que relatórios de exceção permitem que o fornecedor e/ou o cliente do serviço de sessão sejam informados sobre erros menos graves (condições de exceção), que não requerem o encerramento da sessão (ITU-T, 1995).

O protocolo *Handshake* é responsável pelo estabelecimento e restabelecimento das conexões de sessão, implementando, assim, as funcionalidades de estabelecimento de conexão, sincronização e ressincronização da camada de sessão, descritas no Apêndice B. No modelo RM-OSI, o estabelecimento de uma conexão de sessão utiliza as quatro primitivas (*request, indication, response e confirm*). O Quadro 3 faz um comparativo sobre essas

primitivas utilizadas no processo de estabelecimento de conexão do RM-OSI e ações equivalentes no SSL.

PRIMITIVA	RM-OSI	SSL
S_connect. (Request)	Cliente envia pedido de conexão de sessão.	***ClientHello*** Equivalente ao passo 1 da Figura 4.
S_connect. (Indication)	Servidor é sinalizado sobre o pedido de conexão. Executa operações referentes ao pedido.	***ServerHello*** ***ServerKeyExchange*** Equivalente aos passos 2 e 3 da Figura 4.
S_connect. (Response)	Servidor envia resposta ao cliente	***ServerHelloDone*** Equivalente ao passo 4 da Figura 4. Resposta a partir da qual se espera uma ação do cliente.
S_connect. (Confirm)	Cliente recebe resposta do servidor, executa operações e envia confirmação.	***ClientKeyExchange*** ***ChangeCipherSpec*** ***Finished*** Equivalente aos passos 5, 6 e 7 da Figura 4.
S_connect. Confirm	O pedido de conexão foi aceito. Aqui o SSL implementa um passo extra de confirmação.	***ChangeCipherSpec*** ***Finished*** Equivalente aos passos 8 e 9 da Figura 4. A partir daqui existe um identificador de sessão e uma conexão de sessão.

Quadro 3 – Comparativo entre estabelecimento de conexão (RM-OSI x SSL)

O pedido de restabelecimento de conexão de sessão utiliza a funcionalidade de ressincronização. Essa funcionalidade pode ser observada quando, de posse do identificador de sessão, é possível resgatar uma sessão que já existia.

2.6.3 RM-OSI *versus* WAP

Na especificação do protocolo WAP é possível verificar muitas semelhanças com o RM-OSI, como pode ser visto na Figura 6, onde o WSP é visualizado como uma camada separada. No modelo de referência do WAP (Figura 8) é possível visualizar a presença de pontos de acesso aos serviços de sessão (S-SAP), ou seja, as aplicações não implementam serviços de sessão, pois existe uma camada específica para essa funcionalidade.

No WAP a camada de segurança, que fornece criptografia dos dados, está abaixo da camada WSP. Neste caso, comparando-se com o RM-OSI, a ordem está invertida, já que no RM-OSI a camada de apresentação dos dados, onde se enquadram as funções de criptografia, está acima da camada de sessão.

De uma maneira genérica, as seguintes funcionalidades estão presentes nos dois modelos:

a) gerenciamento de sessão (estabelecimento, liberação e retomada de sessão);
b) utilização de dados de capacidade para negociação dos parâmetros de controle;
c) serviços de sessão com conexão e sem conexão; e
d) relatórios de exceção.

O WAP é um padrão criado especificamente para atender à demanda de dispositivos portáteis. Ainda não se sabe se essa tecnologia será largamente utilizada. Entretanto, o WAP foi escolhido como estudo de caso por ser uma tecnologia relativamente nova, em que é possível encontrar várias semelhanças com o modelo de referência RM-OSI.

2.6.4 RM-OSI *versus* SIP

Observa-se que na implementação do SIP existe a necessidade de serviços de sessão para identificar o usuário e manter a sincronização dos dados na Internet. Comparando o SIP com o RM-OSI, em ambos existe gerência de sessão (estabelecimento, modificação e finalização de sessões).

2.6.5 SIP *versus* WAP e SSL

Fazendo uma analogia com a tecnologia WAP, enquanto no WAP o identificador de sessão é o número do telefone portátil, no SIP o identificador é usuário@domínio. Em ambas as tecnologias é necessário encontrar o cliente e oferecer uma sessão lógica persistente, que, no WAP, equivale à troca de área (*roaming*) e, no SIP, equivale à troca de endereço de rede (IP). Nos dois casos o identificador de sessão deve ser único em toda a rede. Assim como não podem existir dois telefones com um mesmo número, também não podem existir dois usuários com o mesmo *login* em um mesmo domínio.

Tanto no SIP quanto no SSL e no WSP, uma sessão possui um identificador e pode conter várias conexões de transporte. Observa-se, então, que a ausência de uma camada de sessão, no projeto inicial da Internet, fez com que as aplicações implementassem os mesmos conceitos de maneiras diferentes, quando estes poderiam ser integrados em uma camada específica e oferecidos em forma de elementos de serviço para as aplicações.

No processo de criação, modificação e término de sessões em redes IP, o protocolo SIP utiliza serviços oferecidos em outros protocolos de aplicação como, por exemplo:

e) tradução de domínios para endereços IP, realizada pelo DNS;
f) localização de usuário, que pode ser realizada pelo LDAP;
g) transporte de dados em tempo real, em que se utiliza o RTP;
h) transmissão de voz com garantia de qualidade, realizada pelo RSVP; e
i) serviço de autenticação, que pode ser realizado pelo Radius.

Essa necessidade torna mais difícil a separação das funcionalidades do SIP em uma camada inferior, pois seu funcionamento precisa de serviços oferecidos pela camada de aplicação. Por exemplo, no SIP a inicialização de sessão não funciona se não existir um serviço de localização e um serviço de resolução de nomes de domínios. Logo, essa separação (se realizada) violaria o modelo de independência de camadas[6].

Como atualmente grande parte do processo de telefonia sobre IP baseia-se na estrutura do SIP, acredita-se que não serão feitos investimentos em adequação desse protocolo à extensão proposta nesta tese, já que o SIP é mais um padrão consolidado. De acordo com Vide (2005), uma linha mais desenvolvida de produtos com a arquitetura SIP já foi disponibilizada pela Cisco, PingTel, 3COM e outros fabricantes. A Microsoft anunciou que não desenvolverá mais ferramentas utilizando o H.323 (como *NetMeeting* e *Exchange Conferencing Server*) e que passará a desenvolver produtos exclusivamente dentro do SIP. Entretanto, existe uma demanda de aplicações futuras que necessitam dos mesmos serviços de sessão. Para essa demanda existem duas possibilidades: continuar se preocupando e reimplementando esses serviços, ou apenas utilizá-los.

2.7 Considerações Finais

Nesse capítulo foram apresentados exemplos de funcionalidades de sessão presentes nas aplicações e protocolos da Internet. Observou-se que a ausência de uma camada de sessão no projeto inicial da Internet impôs, ao longo dos anos, a criação de diversos mecanismos para suprir as necessidades das aplicações que foram surgindo. Sabe-se também que a estrutura da arquitetura da Internet não poderá ser alterada, uma vez que está ativa e funcional. Entretanto, uma demanda de aplicações futuras necessita cada vez mais de um controle de sessão refinado. Separar a funcionalidade de gerência de sessão, presente em todos os estudos de

[6] Definição do RM-OSI: onde a camada “n” utiliza os serviços da camada “n-1” e oferece serviços à camada “n+1”.

caso analisados, certamente contribuirá para a organização da Internet, pois, se não solucionar definitivamente, certamente amenizará o problema de redundância de código.

O próximo capítulo apresenta estudos de casos em que aplicações e protocolos multimídia são analisados no que corresponde à presença dos serviços de sessão. Também são abordados outros problemas atuais relacionados à transmissão do tráfego multimídia.

3 ANÁLISE DAS APLICAÇÕES MULTIMÍDIA

Este capítulo descreve as necessidades e funcionalidades de sessão existentes nas aplicações e protocolos multimídia. O objetivo é realizar um mapeamento, identificando redundâncias nas aplicações existentes e alguns problemas relacionados ao tráfego gerado por esse tipo de aplicação.

A seção 3.1 aborda as necessidades das aplicações, cuja demanda não foi prevista no projeto original da arquitetura da Internet.

A seção 3.2 faz uma breve análise da situação atual da rede no que corresponde à distribuição do tráfego multimídia e do tráfego convencional da Internet.

Os itens 3.3 e 3.4 apresentam os protocolos multimídia destacando a presença destes no desenvolvimento das aplicações.

E, por fim, os itens 3.5 e 3.6 abordam as soluções e propostas destinadas ao funcionamento das aplicações multimídia na rede mundial.

3.1 Necessidades das Aplicações Multimídia

Um requisito importante para se obterem apresentações de áudio e vídeo de boa qualidade é o fornecimento de garantias de desempenho e tempo real, isso tanto no nível de sistemas locais quanto no nível de sistemas de comunicação. Quando a largura de banda, o atraso e a variação de atraso são garantidos, a apresentação multimídia poderá ser feita com qualidade (BORTOLUZZI, 1999).

Segundo Stallings (1999), o tráfego da Internet é dividido em duas categorias: elástico e inelástico, que possuem requisitos e necessidades diferentes. O tráfego elástico pode se ajustar sobre os extensos limites da rede, alterando o retardo e o *throughput*[7], adaptando-se às condições de comunicação da rede. Esse é o tráfego tradicional da Internet, suportado pelas aplicações que utilizam como protocolo de transporte o TCP e o UDP, tais como transferência de arquivo, e-mail eletrônico, *logon* remoto, gerência de rede e acesso à Web. Para esse tipo de tráfego não se gerencia qualidade de serviço (QoS - *Quality of Service*). O tráfego inelástico caracteriza-se por não ser facilmente adaptável, sendo o exemplo principal o tráfego em tempo real, utilizado na transmissão de voz e vídeo. O tráfego inelástico exige qualidade

[7] *Throughput* é a vazão efetiva da transmissão dos dados pelo processamento da informação na fonte, sistema de transmissão e destino.

de serviço, pois requer alto *throughput*, baixo atraso, baixa variação de atraso e baixa perda de pacotes (BORTOLUZZI, 1999).

A fim de fornecer garantias de desempenho para aplicações multimídia, algumas tecnologias, tais como o ATM, implementam o gerenciamento de Qualidade de Serviço. Entretanto, essa não é uma realidade no protocolo da Internet (IPv4), criado há mais de 20 anos, quando a demanda da época não incluía a transmissão de dados multimídia.

Buscando atender à demanda de aplicações que geram tráfego inelástico, esforços foram realizados, adaptando a arquitetura original da Internet ao contexto das necessidades dessa nova geração de aplicações. Entre esses esforços destaca-se o desenvolvimento de alguns protocolos, tais como *Real-Time Transport Protocol* (RTP), *Real Time Transport Control Protocol* (RTCP), *Real Time Streaming Protocol* (RTSP), *Session Description Protocol* (SDP) e outros, que implementam, entre outras funcionalidades, algumas da camada de sessão.

3.2 Tráfego Multimídia da Internet

A popularidade no uso de aplicações que geram fluxo de dados multimídia e de telefonia sobre IP gerou a necessidade de monitorar o volume e as características do tráfego inelástico, particularmente porque o seu comportamento, na presença de um congestionamento da rede, difere do tráfego elástico.

Os próximos itens descrevem estudos realizados sobre a utilização do tráfego multimídia na rede e as alternativas utilizadas na análise desse tipo de tráfego.

3.2.1 Utilização do Tráfego Multimídia

Sripanidklchai, Maggs e Zhang (2004) analisaram, durante um período de três meses, cerca de 70 milhões de requisições originadas em 5.000 URLs distintas, distribuídas sobre 200 países, e apresentaram uma macrovisão do tráfego da rede. O experimento realizou uma coleta de dados (*logs*) em que cada entrada corresponde a uma requisição feita por um cliente a um determinado servidor. Os *logs* armazenados seguiram a seguinte estrutura:

a) identificação do usuário: endereço IP e identificador de player (utilizado pelas aplicações multimídia);
b) objeto de requisição: URL;
c) marcas de temporização: início e duração da sessão, com granularidade em segundos; e

d) estatísticas de desempenho: a média da largura de banda recebida durante uma sessão inteira.

Sobre os dados analisados os pesquisadores concluíram que a maioria da carga diária de *streaming* ao vivo é áudio. Somente 1% das requisições é para fluxo de vídeo. Um pequeno número de eventos, na maioria das vezes programas de áudio, como rádio, são responsáveis por uma gigantesca fração de requisições. Cerca de 50% do grande volume de fluxo de curta duração possui fotografias agrupadas. Com relação ao protocolo de transporte utilizado pelas aplicações multimídia, estatisticamente o protocolo TCP se mostrou dominante na maioria dos domínios AS[8].

Para entender se o uso de um protocolo é específico de uma região, as requisições coletadas foram divididas em domínios AS e países na Tabela 1. Cada região é caracterizada como tráfego dominante TCP ou tráfego dominante UDP, de acordo com o protocolo mais utilizado naquela região. Por exemplo, para as aplicações QuickTime e Real em aproximadamente 50% dos domínios AS, o tráfego dominante é o TCP; já em aproximadamente 25% dos países o tráfego dominante é TCP.

Tabela 1 – Uso dos protocolos de transporte nas aplicações multimídia. Dividido por país e por domínio. Fonte: Adaptada de Sripanidklchai, Maggs e Zhang (2004)

Dividido por Domínio AS	Tráfego dominante UDP	Tráfego dominante TCP
QuickTime	56%	44%
Real	49%	51%
Windows Media	12%	88%
Dividido por País	**Tráfego dominante UDP**	**Tráfego dominante TCP**
Quick Time	75%	25%
Real	72%	28%
Windows Media	2%	98%

Grande parte das versões recentes de aplicações com players de mídia examina automaticamente a rede antes de determinar qual é o melhor protocolo de transporte a ser utilizado. Apesar de o protocolo UDP ser mais adequado para a transmissão multimídia, a configuração de *firewalls*, que filtram o UDP, fez que o TCP também fosse utilizado nesse tipo de transmissão.

[8] Domínio AS corresponde ao serviço internacional de registro de nomes da Internet (IDN – Internationalized Domain Names), disponível em http://www.nic.as/.

3.2.2 Tráfego Inelástico

O comportamento do tráfego inelástico é particularmente diferente do tráfego dominante gerado pelo TCP. Protocolos como SIP, H.323 e RTSP utilizam portas conhecidas e padronizadas para iniciar sessões multimídia. Uma vez que a sessão foi iniciada, esses protocolos negociam outras portas TCP ou UDP dinamicamente, para o controle de tráfego de mídia e para o tráfego de dados de mídia. Esta associação dinâmica de portas dificulta a identificação do tráfego inelástico. Nesse sentido, existem algumas propostas relacionadas a seguir que propõem alternativas para identificar o tráfego inelástico.

Merwe et al. (2000) apresentam o *mmdump*, uma ferramenta que analisa pedaços de mensagens do RTSP e H.323, que são protocolos que realizam controle de sessão. Diferentemente do *tcpdump*, que utiliza um filtro dinâmico de pacotes baseados nas portas de conexão do TCP e UDP, o *mmdump* contém analisadores de gramática específicos para cada protocolo. Esses analisadores agregam fluxos e analisam mensagens de controle extraindo os números das portas, designadas dinamicamente pelo protocolo de sessão multimídia. O filtro dinâmico de pacotes foi modificado para permitir que pacotes associados a uma determinada sessão multimídia sejam capturados.

Os autores do *mmdump* definem uma sessão como uma única instância de interação de um protocolo de controle, como, por exemplo, um cliente RTSP se comunicando com o servidor RTSP ou dois pares que se comunicam pelo protocolo H.323. A ferramenta armazena o estado de cada sessão, contendo informações como início e término da sessão, tempo que ela ficou ativa, tipos de mídia e tráfego associado. A procura de uma sessão (*lookup*) envolve a combinação dos endereços de origem e destino e o número de portas dentro dos pacotes recebidos, que são comparados a valores equivalentes das sessões armazenadas. Para analisar o tráfego dos pacotes, o *mmdump* mantém a sessão ativa: se a procura da sessão tiver sucesso, a sessão é resgatada e usada; senão, uma nova estrutura de sessão é alocada.

O projeto e a implementação do *mmdump* demonstram sua utilidade na visualização do tráfego ativo RTSP e H.323 sobre uma rede comercial. Assim como ele, outros autores também exploram a necessidade de contabilizar o tráfego inelástico em redes comerciais.

Kausar, Briscoe e Crowcroft (1999) dividem o serviço tarifável da rede em duas partes: o serviço da camada de rede e o serviço da camada de sessão. O serviço tarifável da camada de sessão corresponde ao serviço oferecido por um provedor de serviço de aplicação, ou seja, corresponde ao tráfego inelástico gerado pelo impacto das conferências multimídia realizadas

em tempo real na Internet. Os autores destacam a necessidade de contabilização do tráfego inelástico em provedores de acesso particulares e propõem um modelo que utiliza como base de contabilização a participação dos usuários nas sessões multimídia. Neste modelo, a ferramenta SDR (*Session Directory Tool*) foi adaptada para possuir uma opção de pagamento, que deve ser efetuado pelo usuário, a fim de permitir sua participação em conferências multimídia.

A visão dos autores Kausar, Briscoe e Crowcroft (1999) parece ser influenciada pelo ponto de vista comercial das empresas de telecomunicações, uma vez que eles se preocupam mais com alternativas de tarifação do que com a identificação do tráfego.

Observa-se que, por não existir uma categorização de tipos de tráfego na Internet, soluções de contabilização e análise de tráfego inelástico envolvem reprogramação de aplicações ou adaptações específicas, de acordo com o protocolo de sessão multimídia utilizado.

Alterar o código das aplicações é inviável, porque, mesmo que recursos sejam despendidos com esta finalidade, o usuário pode escolher utilizar uma entre diversas aplicações existentes. Outra desvantagem relaciona-se ao fato de elevar à camada de aplicação mais uma funcionalidade, inflando-a ainda mais.

Criar soluções, adaptando um sistema de *parser* para cada um dos protocolos multimídia existentes, é uma tarefa custosa que gera, no mínimo, a programação de dois conceitos de sessão: o primeiro, em nível de camada de aplicação, já utilizado pelos protocolos RTSP e H.323 na execução de suas funcionalidades; e o segundo, em nível de enlace de dados, quando várias conexões de transporte são associadas a uma sessão que observa o comportamento do protocolo de aplicação, identificando todas as portas de transporte que ele utiliza em seu funcionamento.

3.3 Protocolos Multimídia

No contexto de desenvolvimento de aplicações multimídia, observam-se algumas características presentes na maioria das aplicações, como sincronização de áudio e vídeo, técnicas de bufferização e transmissão de dados *multicast*. Entre os protocolos que implementam essas funcionalidades destacam-se o RTP, o RTCP, o RTSP e o SDP.

3.3.1 RTP (Real-Time Transport Protocol)

O protocolo RTP, descrito na RFC 1889 e redefinido na RFC 1890, é utilizado por várias aplicações da Internet em conferências de áudio e vídeo. Seu objetivo principal é fornecer um serviço de transporte para dados com características de tempo real, como áudio e vídeo, satisfazendo, assim, as necessidades da conferência multimídia entre vários usuários. Para isso, ele realiza um controle de sessão (serviço de gerenciamento de sessão) sem oferecer suporte à negociação de parâmetros ou controle dos usuários participantes, porém pode ser utilizado juntamente com outro protocolo da camada de aplicação que realiza esse controle, no caso, o RTCP.

Em redes de pacotes como a Internet, ocorrem ocasionalmente perdas ou reordenação dos pacotes, acrescentando retardos no tempo da transmissão. Para solucionar esse problema, o cabeçalho RTP contém informação de temporização e um número de seqüência que permite que receptores reconstruam a temporização produzida pela origem. Este número de seqüência também pode ser usado como estimativa da quantidade de pacotes perdidos e para determinar a alocação apropriada do pacote, por exemplo, na decodificação de vídeo.

A implementação do RTP não é feita em uma camada separada, mas sim como parte do código da aplicação. O RTP é utilizado com freqüência sobre o *User Datagram Protocol* (UDP), fazendo uso dos serviços de multiplexação e *checksum*[9] que ele fornece.

Apesar de funcionar na camada de aplicação, nota-se que o RTP fornece uma função da camada de transporte de dados e, por isso, é chamado de protocolo de transporte. Ele foi desenvolvido para solucionar o problema de transporte de dados multimídia em uma arquitetura (TCP/IP) não projetada para envio de tráfego inelástico.

3.3.2 Real-Time Transport Control Protocol (RTCP)

O RTCP é responsável pela transmissão periódica de pacotes de controle para todos os participantes de uma sessão multimídia (serviço de sincronização). Ele oferece diagnósticos sobre a qualidade da distribuição dos dados que são utilizados na estimação do atraso de transmissão entre emissores e receptores. Os integrantes de uma sessão multimídia recebem relatórios periódicos (a cada 5 s) contendo o número de seqüência do último dado recebido, o

[9] O UDP faz *checksum* apenas do cabeçalho de dados.

número de pacotes perdidos e uma medida de variação temporal entre recepções e marcas temporais (necessária para verificação do atraso de transmissão entre emissores e receptores).

O controle de congestionamento é feito da seguinte forma: cada participante envia pacotes de controle para todos os outros; portanto, cada um pode, independentemente, observar o número de participantes. Este número é utilizado para o cálculo da taxa na qual os pacotes devem ser enviados.

O RTCP auxilia na sincronização de áudio e vídeo do RTP, realizando a associação de múltiplos fluxos de dados de um determinado participante, dentro de um conjunto de sessões RTP. Também realiza a sincronização entre mídias diferentes. As aplicações que enviaram dados recentemente emitem um relatório que contém informações úteis para a sincronização intermídia: o tempo real e uma correspondente marca temporal. Estes dois valores permitem a sincronização intermídia, como, por exemplo, a sincronização labial.

O RTCP permite a identificação da fonte, pois a mensagem RTCP contém o e-mail do usuário emissor e, opcionalmente, outras informações (nome, endereço, telefone).

Observa-se que o RTCP é responsável pela execução de duas funcionalidades da camada de sessão: a primeira é a funcionalidade de sincronização, pois envia relatórios sobre a qualidade de transmissão dos dados contendo o número de seqüência (equivalente ao ponto de sincronismo) do último dado recebido; e a segunda é o controle de sessão, quando agrupa vários fluxos de dados em sessões.

3.3.3 Real Time Streaming Protocol (RTSP)

Descrito na RFC 2326, é largamente utilizado por servidores de mídia especializados no controle do envio de dados de áudio e vídeo em tempo real. O RTSP permite a interação entre o cliente e o servidor de fluxo contínuo de dados. Se um cliente pode iniciar um fluxo de dados, ele deve ser capaz de parar esse fluxo. Dessa forma, permite-se que o usuário (cliente) tenha controle sobre a reprodução da mídia, por meio dos seguintes comandos:

a) SETUP: induz o servidor a alocar recursos para transmissão de dados e inicia uma sessão RTSP;
b) PLAY E RECORD: inicia uma transmissão de dados alocadas vias SETUP;
c) PAUSE: pára temporariamente uma transmissão de dados sem liberar recursos do servidor; e
d) TEARDOWN: libera recursos associados à transmissão de dados e cessa a sessão RTSP.

Não existe uma noção de conexão de transporte RTSP; em vez disso, o servidor rotula a sessão usando um identificador de sessão RTSP. Dessa forma, uma sessão RTSP não está associada a uma única conexão de transporte TCP. Durante uma sessão RTSP, um cliente RTSP pode abrir e fechar várias conexões de transporte com o servidor, mediante a emissão de requisições RTSP. Existem três tipos de conexões RTSP: conexões de transporte persistentes, usadas por várias transações de requisições e respostas; uma conexão para cada transação de requisição e resposta; e no modo sem conexão, que utiliza, nesse caso, o UDP. Tanto para conexões UDP quanto para conexões TCP, a porta *default* é a 554.

Em geral, o estabelecimento do fluxo de dados é feito pelo RTP, mas a operação do RTSP não depende de um mecanismo usado para envio de média contínua. Sua sintaxe é semelhante à sintaxe do HTTP 1.1.

Observa-se que o RTSP implementa funções de controle de sessão já que, ao contrário do HTTP, ele mantém o estado da conexão ao utilizar um identificador de sessão associado a uma ou mais conexões de transporte.

3.3.4 Session Description Protocol (SDP)

O SDP foi definido na RFC 2327, que descreve o conteúdo das sessões, incluindo telefone, rádio e aplicações multimídia da Internet. Ele inclui informações sobre o fluxo de mídia[10], endereços (*unicast* e *multicast*), portas de controle (utiliza o UDP), tipos de *payload* (formato utilizado durante a sessão), tempo de início e parada (usado em programações de televisão e rádio), e origem, utilizado no caso de ocorrência de dificuldades técnicas que o impeçam de obter informações de contato.

O SDP fornece a capacidade de descrever um conteúdo multimídia, porém faltam mecanismos para que as duas partes concordem com os parâmetros que serão utilizados na conferência. A RFC 3264 contorna essa falta por meio da definição de um modelo simples de oferta/resposta, em que as duas partes enviam mensagens SDP para alcançar um acordo sobre a natureza do conteúdo multimídia distribuído.

O funcionamento dos protocolos de anúncio de sessão, tais como o SDP, pode ser comparado ao conceito de atividade da camada de sessão, onde as primitivas de início e fim de atividade sinalizariam o início e fim de uma sessão multimídia. A primitiva

[10] Uma sessão pode conter fluxos de áudio, vídeo e dados, além de controle e tipos de fluxos de aplicações, tais como o formato *Multipurpose Internet Mail Extensions* (MIME).

s_activity_resume, por exemplo, conteria o resumo das atividades correntes, ou seja, quais sessões estão ativas. Dessa forma, o usuário pode escolher em qual sessão deseja participar.

Feita uma avaliação sobre o funcionamento dos principais protocolos multimídia, o próximo item avalia a freqüência e utilização desses protocolos nas aplicações desenvolvidas.

3.4 Utilização dos Protocolos Multimídia

Atualmente, existe uma grande demanda de aplicações multimídia na Internet, na qual se observa, com freqüência, a presença dos protocolos descritos anteriormente.

O Quadro 4 apresenta dados sobre as aplicações que implementam o protocolo RTSP, combinado com outros protocolos (RTSP, 2003).

Organização	Nome da Aplicação	Sistema Operacional	Tipo	Protocolos	Código Fonte
Apple	QuickTime Streaming Server	MacOs Linux	Servidor	SDP, RTP, RTSP	Completo
Apple	Quick Time 4	MacOs, Windows	Cliente	SDP, RTSP	Não disponível
Columbia University	Rtspd	NT, Solaris	Servidor	SDP, RTSP	Disponível
Live.com	openRTSP	Linux x.86, FreeBSD, Solaris SPARC, Windows	Cliente	SDP, RTSP	Disponível
Oracle Corporation	Oracle Video Server	Windows, Solaris, IRIX, TruUnix, Linux, Transit, VxWorks, OS-9	Cliente, Servidor	SDP, RTSP	Código C cliente disponível com licença
Real Networks	Real Networks	Windows FreeBSD, Linux	Servidor	SDP, RTSL, RTSP	Código C disponível
Real Networks	Real Server G2	Windows, FreeBSD, Linux, Solaris	Servidor	SDP, SMIL, RTP, RTSP	Não
Vovida	RTSP	Linux	Pilha Servidor	RTSP	Código livre
Microsoft	Windows Media Services	Windows	Servidor	RTSP, MMS	Não

Quadro 4 – Implementações RTSP. Fonte: Adaptada de: CS at Columbia University(2003)

Em algumas aplicações de código livre, como, por exemplo, o DDS (DarwinStreamingSrc[11]) e o LiveCaster[12], o comportamento usual é implementar esses protocolos como parte de seu código-fonte. Isso é feito porque não existe uma biblioteca ou camada intermediária que trate apenas de serviços de controle de sessão. O Quadro 5 utiliza como exemplo trechos dos arquivos RTSPClient.h do DSS e RTSPClient.hh do LiveCaster, em que é possível observar que, embora ambas as aplicações ofereçam o mesmo tipo de serviço, existem diferenças de implementações em seu código-fonte.

Aplicação / Arquivo	Implementação de classes	Trecho retirado do início da função
DDS File: RTSPClient.h	class Authenticator class BasicAuth : public Authenticator class DigestAuth : public Authenticator class AuthParser class RTSPClient	#ifndef __RTSP_CLIENT_H__ #define __RTSP_CLIENT_H__ #include "OSHeaders.h" #include "StrPtrLen.h" #include "TCPSocket.h" #include "ClientSocket.h" #include "RTPMetaInfoPacket.h"
LiveCaster File: RTSPClient.hh	class RTSPClient	#ifndef _RTSP_CLIENT_HH #define _RTSP_CLIENT_HH #ifndef _MEDIA_SESSION_HH #include "MediaSession.hh" #endif #ifndef _NET_ADDRESS_HH #include "NetAddress.hh" #endif

Quadro 5 – Redundância de código

A Figura 10(a) é uma representação de como é realizada a implementação dos protocolos multimídia dentro das aplicações, e a Figura 10(b) apresenta uma alternativa de organização que solucionaria o problema de redundância de código, separando funcionalidades de sessão em bibliotecas específicas.

[11] DDS é a versão de código aberto da tecnologia QuickTime Streaming Server da Apple que permite enviar vários *streams* de mídia (áudio, vídeo e dados) para clientes através da Internet. Utiliza o C como linguagem de programação.

[12] LiveCaster é uma aplicação, implementada em C++, cuja funcionalidade principal é realizar *streams* de áudio multicast para um número potencialmente ilimitado de clientes.

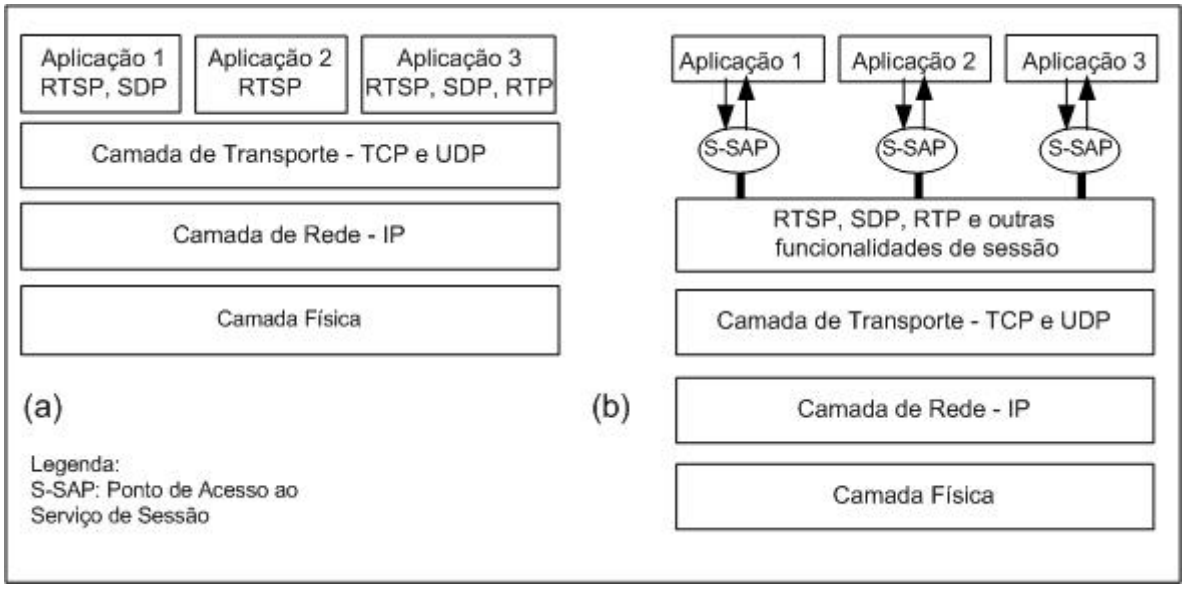


Figura 10 – Implementação dos protocolos multimídia: (a) Com código fonte dentro das aplicações; (b) Acesso às funcionalidades via ponto de acesso de serviço

Além da redundância de funcionalidades encontradas dentro das aplicações multimídia, observou-se que a técnica de bufferização também está presente na maioria dessas aplicações. Uma análise sobre o funcionamento desses buffers é apresentada a seguir.

3.5 Técnica de Bufferização

Buscando reduzir a variação de atraso para melhorar a apresentação do áudio e do vídeo na transmissão de dados em tempo real, é comum utilizar a técnica de bufferização, que consiste em armazenar os dados em uma área de armazenamento temporário (buffer*)* antes da reprodução. Se o servidor funcionar em velocidade de reprodução normal, os dados que chegarem dele serão anexados no final do buffer, e o reprodutor consumirá os dados do início do buffer. Buscando evitar intervalos na transmissão do áudio, o servidor envia dados com maior rapidez. Porém, se o reprodutor não consumir esses dados em tempo, pode ocorrer estouro de buffer, e os dados podem ser descartados. Reprodutores de mídia utilizam um buffer com dois limites, um inferior e outro superior. Dessa forma, o servidor pode acelerar ou interromper temporariamente o envio de dados para o cliente, mantendo o buffer com dados em nível suficiente para a reprodução da mídia.

A abordagem mais utilizada para a remoção dessa variação de atraso é o uso de buffers do tipo *First-In First-Out* (FIFO) no destino, antes da apresentação. Esta técnica é chamada de técnica de bufferização e pode ser visualizada na Figura 11. Nela, à medida que os pacotes

chegam (em uma taxa variada), eles são colocados no buffer; o dispositivo de apresentação retira amostragens do buffer a uma taxa fixa [LU, 1996].

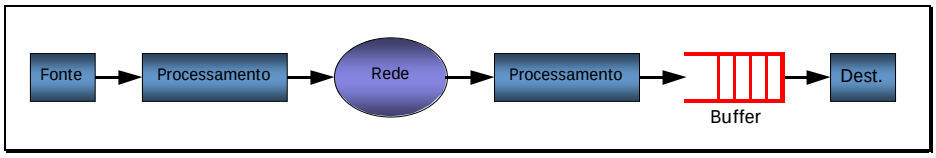


Figura 11 – Técnica de bufferização. Fonte: Adaptada de Lu (1996)

Aplicações multimídia como radiodifusão, ensino a distância, programas de televisão coorporativos e apresentação de videoclips possuem clientes assíncronos, isto é, que assistem aos vídeos em diferentes instantes de tempo. Por exemplo, um servidor de videoclip iniciou duas horas de transmissão de dados para um cliente, e um segundo cliente requisitou a mesma transmissão dez minutos mais tarde. Para cada cliente será aberto um canal de transmissão de dados distinto. Nesse tipo de aplicação, para se ter baixa latência, utiliza-se um grande número de canais de transmissão, o que aumenta a carga no servidor e na rede. Uma maneira de minimizar esse problema é distribuir o fluxo de mídia entre múltiplos clientes assíncronos, sem introduzir retardo em qualquer um desses clientes (SEN; TOWSLEY, 1999).

Sen e Towsley (1999) propõem duas técnicas que permitem o compartilhamento dos dados existentes na transmissão do mesmo vídeo. A primeira, denominada *Periodic Buffer Reuse* (PBR), é um algoritmo que maximiza a quantia de dados que um cliente pode restaurar a partir do início de uma transmissão. A segunda, *Greedy Buffer Reuse* (GBR), é um algoritmo que permite que servidores utilizem canais simultâneos na transmissão de dados, o que reduz consideravelmente o *overhead*. Tais algoritmos podem ser utilizados como base na implementação de controle de buffers (SEN; TOWSLEY, 1999).

Além dos buffers utilizados na camada de aplicação, existem os buffers utilizados pelo TCP no transporte dos dados. Este trabalho está contextualizado nas funcionalidades da camada de sessão. Entretanto, buscando mapear a demanda de aplicações multimídia, também se encontram propostas de alterações na camada de transporte, que serão abordadas a seguir.

3.6 Soluções para Aplicações Multimídia

Esta seção apresenta uma seleção de propostas atualizadas apresentadas em periódicos em que se buscam soluções aos problemas encontrados na transmissão de dados multimídia.

Além das funcionalidades de sincronização, atividade e controle de sessão presentes nos protocolos padronizados pelo IETF, e descritos na seção 3.3, existem outras propostas em desenvolvimento em que também é possível identificar funcionalidades da camada de sessão.

Sousa e Freitas (1998) implementaram um *framework* que oferece suporte ao desenvolvimento de aplicações tolerantes a falhas. No cenário de operação, quando um usuário remoto deseja juntar-se a uma sessão multimídia, ele executa um *plug-in* em seu servidor WWW cliente, que ativará o controle responsável pela sua participação, abrindo canais RTP e RTCP entre todos os clientes que participam da sessão multimídia. A arquitetura do *plug-in* implementa quatro módulos que utilizam os protocolos RTP e RTCP para envio de dados *unicast* e *multicast*. Esses módulos são responsáveis pela transmissão e recepção dos dados, controle de QoS, configuração do protocolo, identificação e sincronização.

Simon, Sood e Mundur (2000) propõem uma arquitetura para aplicações multimídia distribuídas (*Distributed Multimedia Applications* – DMS) que consiste de três subsistemas distintos: um produtor de dados multimídia sincronizados em tempo real; uma rede de comunicação; e um consumidor de informação de dados sincronizados em tempo real. A arquitetura DMS garante que a sincronização é mantida, reforçando a alocação de recursos e a admissão de políticas de controle dentro dos vários subsistemas. O DMS aceita a aquisição e executa vários algoritmos de admissão de controle, para cada um dos subsistemas. Isso significa que os recursos devem estar disponíveis em cada subsistema antes que uma nova requisição seja admitida dentro dos vários subsistemas.

Chen, Bodenheimer e Barnes (2003) propõem um esquema de transmissão híbrida, buscando atender à demanda de aplicações de jogos que possuem imagens tridimensionais. Este esquema híbrido sugere uma mescla, onde uma porcentagem dos dados é enviada via TCP e outra porcentagem é enviada via UDP. Dessa forma, é possível calibrar a aplicação de acordo com o estado do canal de comunicação de dados, obtendo resultados significativos na qualidade final da imagem transmitida.

Wang (2003) propôs um novo tipo de socket adequado ao transporte de dados de aplicações multimídia. O UDTCP socket foi implementado no kernel do sistema operacional FreeBSD 4.8 e implementa propriedades dos sockets TCP e UDP, que são adequadas para o transporte de dados onde o atraso é sensitivo. Porém, ele não implementa as propriedades não adequadas a esse tipo de aplicações, deixando os outros tipos de aplicação a descoberto. No transporte dos dados multimídia, a taxa de envio precisa ser regulada de acordo com a latência da rede e, para isso, utiliza-se o controle de congestionamento do TCP. Entretanto, o controle de erros não é desejável, já que na retransmissão dos pacotes perdidos os pacotes reenviados podem chegar muito tarde para serem utilizados pela aplicação. A implementação do UDTCP baseou-se na alteração do código-fonte do TCP. Para converter o TCP ao UDTCP, foram inseridas 43 linhas de código C padrão, fazendo com que o UDTCP fornecesse o controle de

congestionamento do TCP, mas não implementasse o controle de erros. A implementação dentro do kernel fez com que a aplicação não precisasse se preocupar em implementar esquemas de controle de congestionamento, que o autor denominou de TCP-*like*.

O Instituto de Tecnologia da Califórnia desenvolveu um sistema chamado Fast TCP, que tem como característica-chave o fato de poder funcionar na mesma infra-estrutura da Internet de hoje. O TCP divide arquivos grandes em pequenos pacotes de cerca de 1.500 bytes, e se a confirmação não chegar o remetente transmite o mesmo pacote com metade da velocidade anterior, até conseguir. Dessa forma, pequenos ruídos na linha podem tornar a transmissão muito lenta. O Fast TCP usa o mesmo tamanho de pacotes que o TCP comum. A diferença está no software e no hardware do computador remetente, que mede constantemente o tempo que os pacotes levam para chegar ao destino e o tempo de retorno da confirmação. Com isso é possível analisar a conexão e identificar prováveis perdas de pacotes. O software do Fast TCP utiliza essa informação para calcular o maior índice de dados que a conexão pode suportar sem a ocorrência de perdas. Ao combinar dez sistemas de Fast TCP, pesquisadores conseguiram velocidades de transmissão superiores a 8,6 gigabits por segundo, ou seja, mais de 6 mil vezes a capacidades das conexões comuns de banda larga (CALTECH, 2005).

3.7 Considerações Finais

A transmissão de dados em tempo real agrega uma série de propostas em busca da solução de seus problemas. Entre essas alternativas encontram-se padrões como os protocolos multimídia descritos no item 3.3, soluções específicas implementadas dentro das aplicações e alterações dos protocolos de transporte da Internet, todos em busca de alternativas que reduzam o atraso na transmissão de dados multimídia. Portanto, seja qual for o modelo de sessão proposto, ele deve preservar o que já existe. Durante os mais de 20 anos da Internet muitas aplicações foram criadas, e acreditar que essas aplicações serão alteradas para se adaptar a um novo modelo é ilusão.

Até agora, apresentaram-se os resultados da pesquisa exploratória realizada que teve como estudos de caso algumas aplicações, padrões e protocolos da Internet. Uma análise comparativa permitiu constatar a presença de sessão nesta arquitetura. O próximo capítulo apresenta o objeto desta tese, a proposta de uma extensão que contemple as funcionalidades de sessão que se pretende adequar à arquitetura da Internet.

4 A CAMADA DE SESSÃO PROPOSTA

Como se pôde constatar nos Capítulos 2 e 3, o controle de sessão existe e é importante para o desenvolvimento das aplicações. Aplicações multimídia em tempo real são exemplos dessa necessidade. De fato, os mecanismos de sessão são implementados dentro de vários protocolos e aplicações, e poderiam ser tratados por uma camada específica, atendendo às necessidades das aplicações.

O objeto de estudo desta tese não busca substituir as implementações, e, sim, propor uma extensão à arquitetura da Internet que atenda à demanda de futuras possíveis aplicações e que possa coexistir com os padrões já definidos. Neste sentido, esse capítulo descreve a camada de sessão proposta.

4.1 Visão Geral

A extensão proposta separa as funcionalidades de sessão em uma camada específica, localizada entre as camadas de transporte e de aplicação. Ela adapta as funcionalidades descritas na quinta camada do modelo RM-OSI ao contexto da Internet, contempla algumas necessidades implementadas dentro das aplicações multimídia e reserva um espaço para atender a demandas ainda não identificadas.

Não se pretende forçar a reprogramação das aplicações existentes, mas, sim, oferecer serviços que possam auxiliar na organização e no desenvolvimento de novas aplicações, gerando um código mais limpo e integrando funcionalidades, sem redundâncias, em uma camada específica. Dessa forma, a extensão proposta deve oferecer serviços de controle de sessão para as aplicações e manter a flexibilidade da arquitetura da Internet. Esta flexibilidade é mantida com o uso de mecanismos de *pass-through*.

A pesquisa bibliográfica realizada avaliou as necessidades e redundâncias das aplicações multimídia referentes às funcionalidades de sessão, conforme apresentado no Capítulo 3. Considerando-se essas necessidades, chegou-se à definição de um modelo dividido em cinco componentes, que pode ser visualizado na Figura 12. O Quadro 6 apresenta uma visão geral desses componentes, que serão abordados com mais detalhes neste capítulo.

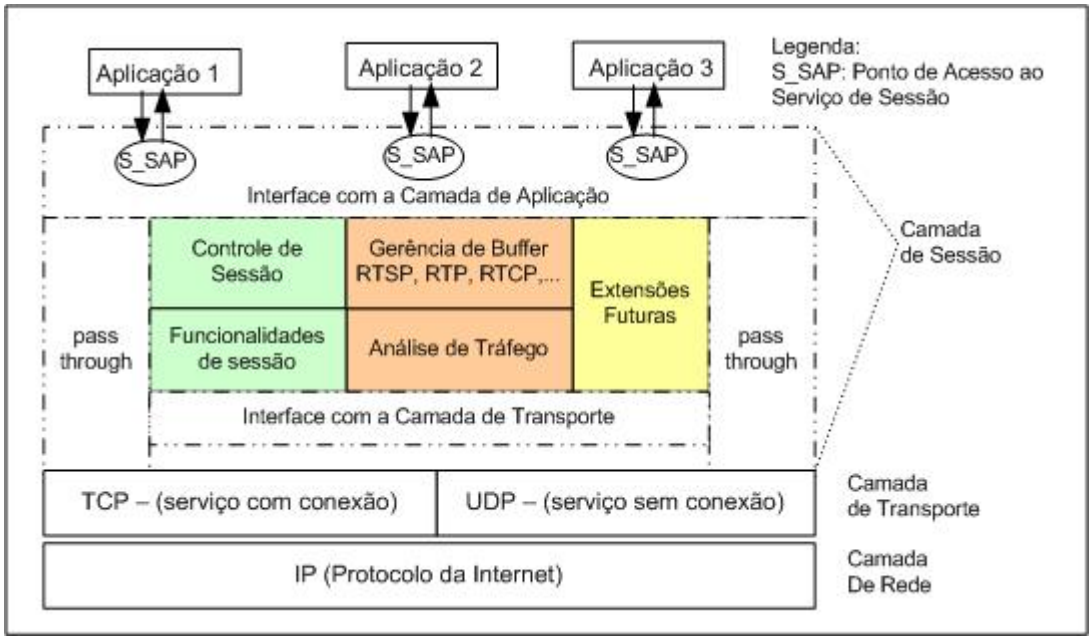


Figura 12 – Componentes da camada de sessão proposta

COMPONENTE	DESCRIÇÃO
Controle de Sessão (CS)	Responsável pelos serviços de estabelecimento, finalização e recuperação de conexão de sessão. Responsável por gerar um identificador de sessão.
Funções de Sessão (FS)	Implementa um subconjunto das funções da camada de sessão do modelo RM-OSI.
Análise de Tráfego em Nível de Sessão (ATs)	Gera estatística sobre o tráfego de dados, identifica e agrupa fluxos de tráfego.
Gerência de Buffers em Nível de Sessão (GBs)	Implementa algoritmos e protocolos de controle de buffers, utilizados na transmissão de dados em tempo real. Realiza o controle do buffer das entidades pares da camada de sessão. Realiza o controle do buffer de player.
Extensões Futuras em Nível de Sessão (EFs)	Prevê portabilidade para outras camadas e futuros serviços de sessão.

Quadro 6 – Descrição dos componentes da camada de sessão

4.2 Elementos de Comunicação entre Camadas

A camada de sessão proposta se comunica com a camada de transporte através de um dos quatro tipos de serviço:

a) serviço com conexão de sessão confirmado (ou sessão confirmada): usa os serviços do TCP na camada de transporte;

b) serviço com conexão de sessão não confirmado (sessão não confirmada): usa os serviços do UDP na camada de transporte;
c) *pass-througth* TCP: não fornece serviço de sessão; apenas oferece uma passagem direta da aplicação para o protocolo TCP da camada de transporte; e
d) *pass-through* UDP: não fornece serviço de sessão; apenas oferece passagem direta da aplicação para o protocolo UDP da camada de transporte.

O modelo da camada de sessão pressupõe que, para oferecer serviços de sessão, deve existir uma conexão de sessão ativa, por onde passam dados de controle. A existência dessa conexão de sessão está associada a um identificador de sessão (id_session). Assim, a exemplo do protocolo SSL, a partir do momento que existe um id_session, existe uma sessão.

Existindo um identificador de sessão (id_session), o envio de dados pode ser feito por um serviço de transporte confirmado ou não confirmado, de acordo com a necessidade da aplicação; exceto para os dados de capacidade. Dados de capacidade são utilizados na transferência de mensagens de controle da camada de sessão e utilizam sempre um serviço de transporte confirmado, porque sua entrega deve ser garantida, facilitando o funcionamento da extensão proposta. Se na arquitetura da Internet existe um serviço de transporte confirmado que garante a entrega dos dados (TCP), então ele deve ser utilizado para simplificar a constituição da extensão, uma vez que não se pretende se preocupar com funções da camada de transporte em nível de camada de sessão.

Como visto no Capítulo 3, aplicações multimídia podem desejar enviar dados sobre o UDP. Para estas aplicações foi criado o serviço de sessão não confirmado. Entretanto, demais funcionalidades da camada de sessão podem exigir um serviço confirmado, como, por exemplo, as mensagens de controle. Dessa forma, associadas a um mesmo identificador de sessão, podem existir várias conexões de transporte, algumas sobre o TCP (serviço de sessão confirmado) e outras pelo UDP (serviço de sessão não confirmado).

Como todo controle é feito por meio de um serviço confirmado (TCP) e o protocolo UDP é mais rápido que o TCP, pode ocorrer que os pacotes UDP cheguem antes dos pacotes de controle (transmitidos através de um serviço confirmado).

Supondo que uma determinada aplicação necessite ressincronizar os dados enviados por meio de um serviço não confirmado, se a sessão de controle estiver ativa, a PDU de sincronização será enviada por ela; porém, se ela não estiver ativa, terá que ser retomada, para então realizar o pedido de sincronização. Essa retomada de sessão gera um overhead em nível de camada de transporte, o que deve ser evitado, principalmente se um serviço de transporte de dados não confirmado (UDP) está sendo utilizado. Neste caso, as mensagens de controle de

sessão podem não refletir a atual situação da transmissão dos dados, ocasionando descarte dos pacotes (UDP) recebidos recentemente. Por exemplo, se a marcação de um ponto de sincronismo demorar mais do que o esperado (devido ao *overhead* gerado pelo estabelecimento da conexão), o envio de dados sobre o serviço não confirmado continuará, e todos os dados recebidos depois do ponto de sincronismo terão que ser retransmitidos.

Portanto, visando reduzir o *overhead* existente no estabelecimento de uma conexão TCP, a conexão de controle deve manter-se ativa, só sendo desconectada após um longo tempo de inatividade, quando o temporizador expirar.

4.2.1 Considerações Sobre o Uso do UDP

Observando a implementação de alguns protocolos da Internet, como, por exemplo, NFS (Network File System), verificou-se que é possível ter um serviço confirmado sobre o protocolo UDP. O NFS implementa seus próprios mecanismos de confirmação e controle de erros, descritos na RFC 1094.

O escopo deste trabalho não considera tal possibilidade, uma vez que garantir a entrega das mensagens é função da camada de transporte, e não da camada de sessão.

Como reforço a essa decisão, pode-se citar o protocolo WSP, descrito no Capítulo 2, que utiliza na camada de transporte um modo com conexão e confirmado (WTP) para a maioria de suas funcionalidades.

4.3 Componente de Controle de Sessão (CS)

O componente Controle de Sessão é responsável por estabelecer, finalizar e resgatar sessões existentes. Ele também oferece os serviços de relatórios de anomalias, que permite sinalizar a aplicação sobre o estado da conexão de sessão. As primitivas de serviço utilizadas são visualizadas no Quadro 7.

Somente depois de estabelecer uma sessão e negociar os parâmetros desta sessão, os demais serviços da camada estarão disponíveis para as aplicações, com exceção dos serviços de *pass-through*, que dispensam a existência de uma conexão de sessão.

PRIMITIVA	R	I	Rs	C	FUNÇÃO
Serviço com Conexão (sobre TCP)					
S_CONNECT	•	•	•	•	Estabelecimento de conexão
S_CONNECT_M	•	•	•	•	Estabelecimento de conexão multiponto
S_RELEASE	•	•	•	•	Liberação negociada de conexão
S_U_ABORT	•	•			Liberação abrupta (usuário)
S_P_ABORT		•			Liberação abrupta (fornecedor)
S_U_EXCEPTION_REPORT	•	•			Relatório de anomalia (usuário)
S_P_EXCEPTION_REPORT		•			Relatório de anomalia (fornecedor)
S_ MESSAGE_REPORT	•	•			Mensagem imprópria
S_ AUT_REPORT	•	•			Falha na autenticação do id_session

Quadro 7 – Primitivas de serviço do componente CS

De forma semelhante à adotada na arquitetura OSI, no modelo proposto também são utilizadas as primitivas de serviço (*request, indication, response* e *confirm*) descritas no Apêndice C.2.1.

Além das primitivas do RM-OSI, a extensão proposta oferece um serviço de conexão multiponto (CSMu). Para oferecer esse serviço, foi necessário adicionar uma primitiva de serviço s_connect_m, pois a conexão de sessão multiponto possui características diferentes da conexão de sessão normal. Essa funcionalidade será descrita em mais detalhes no item 4.3.5.

A idéia de agrupar as primitivas de serviço de controle de sessão em um componente específico baseou-se nas seguintes considerações:

a) nenhuma funcionalidade de sessão poderá ser requisitada se não existir uma conexão de sessão preestabelecida;
b) separar o controle de sessão das demais funcionalidades facilita a inclusão de futuras extensões ao modelo;
c) o modelo de referência RM-OSI separa na unidade funcional Kernel as funções básicas de estabelecimento e encerramento de conexão; e
d) os relatórios emitidos para a camada de aplicação refletem o estado da conexão de sessão (ativa, inativa, tempo expirado (*time-out*), problemas com a conexão de sessão no lado usuário ou no lado servidor).

Logo, o que é regra comum para todas as funcionalidades de sessão, deve ser separado em um componente distinto. Todas as informações de controle da camada são tratadas pelo componente CS, que armazena em um registro as informações da sessão.

A utilização de um identificador de sessão abstrai o conceito de portas de conexão (portas TCP e UDP). No momento em que existe um identificador de sessão, mesmo que a conexão de transporte caia, a conexão de sessão continua ativa e pode ser resgatada enquanto não expirar o seu período de atividade. O período de atividade, ou tempo de expiração, é

determinado pela camada de aplicação, sendo passado como parâmetro no momento em que ela faz um pedido de estabelecimento de sessão (s_connect).

4.3.1 Estabelecimento de Conexão de Sessão

A fase de estabelecimento de uma conexão de sessão (ou simplesmente, estabelecimento de sessão) ocorre entre dois usuários participantes e precisa de um protocolo de transporte confiável. A sessão é bidirecional, ou seja, qualquer usuário participante pode iniciá-la. Para facilitar o entendimento do processo de estabelecimento de conexão de sessão considera-se cliente a entidade par que iniciou a sessão e servidor a entidade par que recebeu o pedido de estabelecimento de sessão.

O pedido de estabelecimento de uma sessão é sempre feito pelo cliente; entretanto, o servidor é responsável pelo cálculo e fornecimento do identificador de sessão. Essa característica foi adotada após observar o comportamento dos protocolos da Internet (SSL, descritos no Capítulo 2).

Além do id_session, outros parâmetros são negociados pelo protocolo no momento do estabelecimento da sessão. Esses parâmetros estão descritos no Quadro 8.

PARÂMETRO	DESCRIÇÃO
id_session	Identificador de Sessão. É um valor único definido pela entidade de sessão do servidor durante o processo de estabelecimento e negociação de classes de serviço de sessão.
id_serviçe	Identifica a primitiva de serviço de sessão.
id_classes	Indica as classes de serviço de sessão negociadas, descritas nos componentes CS, FS e GB.
id_categoria	Indica a categoria de tráfego utilizada.

Quadro 8 – Parâmetros de negociação

O campo id_categoria deve ser preenchido pela aplicação e permite identificar o tipo de tráfego que será transmitido pela rede. Essa funcionalidade de análise de tráfego está descrita em detalhes no item 4.6.

O cliente solicita o estabelecimento de uma sessão através da primitiva s_connect.*request*[13] e envia para o servidor uma PDU contendo IP de origem, IP de destino, porta de comunicação de origem e porta de comunicação de destino[14].

[13] Primitiva de serviço do componente CS (Quadro 7).

O servidor executa as seguintes atividades:

a) gera um identificador de sessão;
b) armazena em uma Tabela local informações como id_session, IP de origem, IP de destino, porta de comunicação de origem, porta de comunicação de destino e tempo de expiração da sessão; e
c) envia para o cliente o identificador de sessão, junto com parâmetros que identificam as classes de serviço oferecidas.

O cliente recebe as informações, armazena os dados e comunica a aplicação.

A Figura 13 descreve o funcionamento da máquina de estados do estabelecimento de uma conexão de sessão, descrita com mais detalhes no Apêndice A.

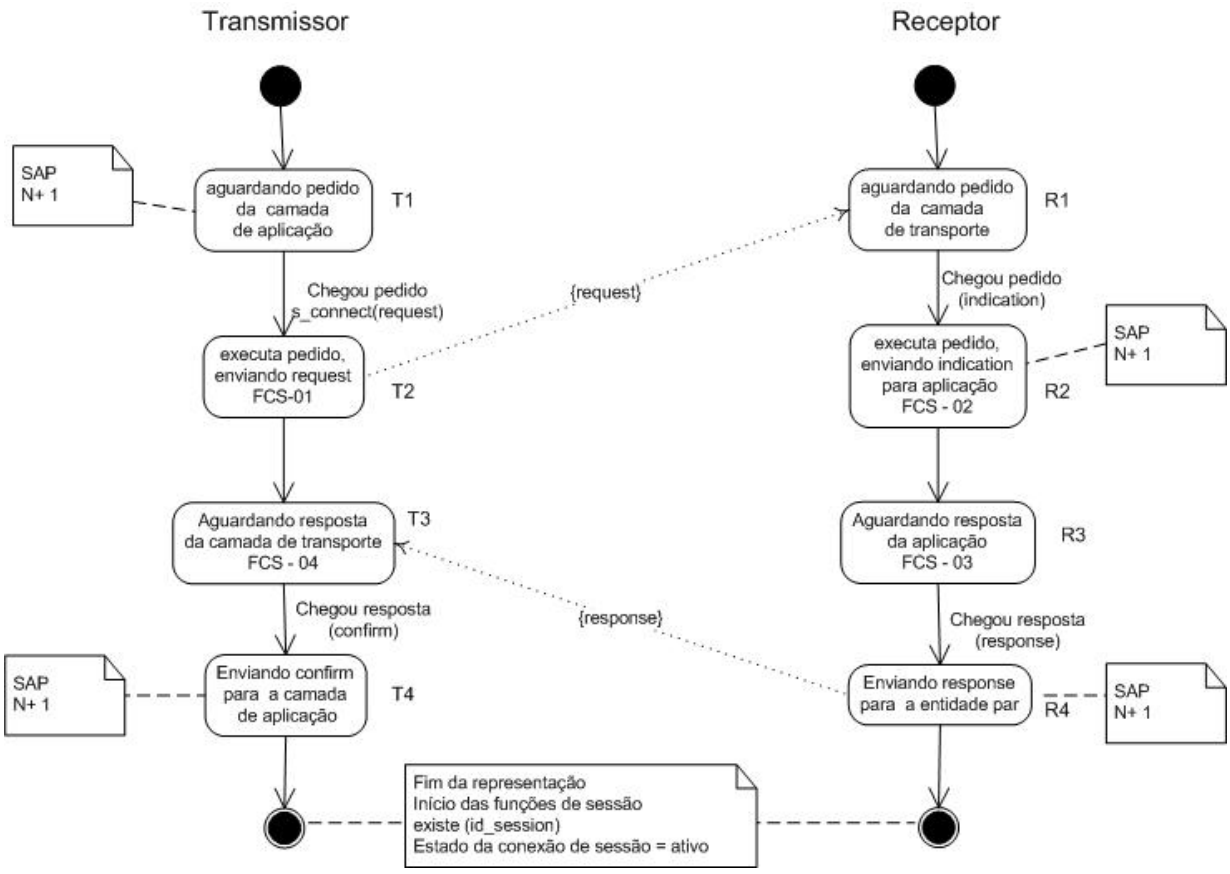


Figura 13 – Estabelecimento de uma sessão

[14] Existem outros parâmetros que são utilizados na primitiva s_connect, de acordo com a classe de serviço que se pretende utilizar.

Associado a um identificador de sessão está seu tempo de expiração. Esse valor é um parâmetro enviado pela aplicação na primitiva s_connect. Caso a aplicação não defina um tempo de expiração, assume-se o valor zero. Se o tempo de expiração for igual a zero, quando a sessão for interrompida (porque caiu a conexão de transporte ou outro motivo), ela não poderá ser resgatada, havendo, assim, a necessidade de se iniciar um novo processo de estabelecimento de conexão de sessão para cada operação realizada.

Observando o comportamento dos protocolos de transporte da Internet, verificou-se a necessidade de reconhecer a legitimidade do cliente. Por exemplo, se o cliente estiver em uma rede com NAT, não existem garantias de que a porta de conexão utilizada será sempre a mesma; pelo contrário, ela com certeza irá mudar caso a conexão cair. Para garantir a legitimidade do cliente, garantindo o funcionamento do protocolo, o id_session deve ser passível de verificação. Essa necessidade de verificação pode ser negociada, podendo ser realizada de duas formas:

a) a cada nova conexão de transporte e toda vez que a sessão for resgatada; ou
b) a cada PDU de dados ou controle.

4.3.2 Liberação de Conexão de Sessão

Semelhante ao serviço definido pelo modelo RM-OSI, a liberação de uma sessão pode ocorrer por iniciativa do usuário ou do fornecedor do serviço de sessão. A máquina de estados apresentada na Figura 14, abaixo, ilustra as três possibilidades de liberação de sessão. Mais detalhes estão descritos no Apêndice A.

Se o serviço de sessão for liberado de forma negociada (utilizando a primitiva s_release), a camada de sessão retira as informações do registro da tabela de sessões. Se o serviço for liberado de forma abrupta, o registro da sessão permanece na tabela de sessões enquanto o período de atividade não extrapolar ou se a sessão for resgatada pelo cliente.

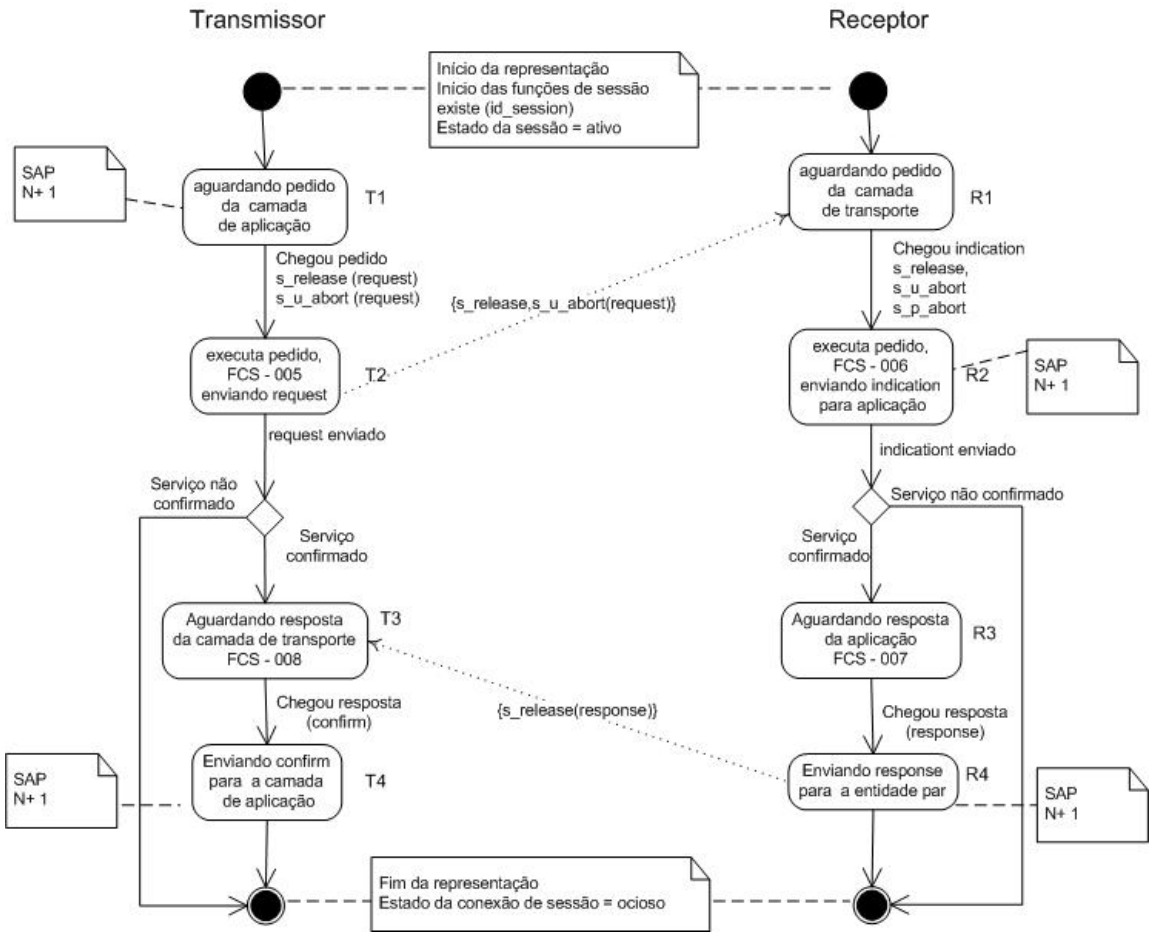


Figura 14 – Finalização de uma sessão

4.3.3 A Recuperação de uma Sessão Existente

Não existem primitivas de serviço para acessar o serviço de recuperação da sessão. Esse serviço é automático e faz parte da programação da camada de sessão. Se uma sessão não foi finalizada de uma maneira negociada e não expirou seu período de atividade, os registros desta sessão ainda estão armazenados na tabela de sessões e ela pode ser resgatada. Para a sessão ser resgatada, basta que o cliente encaminhe ao servidor uma PDU contendo o identificador da sessão (id_session) e o endereço IP (do cliente). O servidor irá ativar novamente a sessão, abrindo novas portas de comunicação para ela e restabelecendo o período de atividade definido pela aplicação.

Assim como no SSL, descrito no Capítulo 2, no modelo proposto a recuperação de uma sessão agiliza o processo de restabelecimento de comunicação, porque não é necessário

recalcular o identificador de sessão sempre que houver uma falha de comunicação na conexão de transporte.

4.3.4 Relatório de Anomalias

Este serviço permite informar a camada de aplicação sobre as anomalias que podem ocorrer na sessão de controle do componente CS. Além das primitivas conhecidas pelo RM-OSI, mais algumas foram inseridas, tendo por base o protocolo de alerta do SSL (THOMAS, 2000).

Os seguintes eventos são notificados:

a) a conexão de transporte foi fechada (pelo cliente ou pelo servidor);
b) uma mensagem imprópria foi recebida (uma classe de serviço que não foi negociada ou não pode ser atendida pelo protocolo); e
c) falha na autenticação do id_session, pelo menos uma das informações do registro (id_session ou IP) é inválida.

4.3.5 Controle de Sessão Multiponto

A camada de sessão permite a utilização de uma conexão de sessão multiponto (CSMu). Nesse serviço, o servidor gera um identificador de sessão multiponto que será utilizado por todos os clientes participantes. O cliente que deseja participar conecta-se ao servidor através da primitiva s_connect_m, e autoriza que outros clientes utilizem seu recurso de buffer. Neste caso, toda mensagem que chega do servidor para o cliente é armazenada em um buffer local, de onde a aplicação irá consumir os dados. Ao utilizar a primitiva s_connect_m o cliente notifica ao servidor sobre a possibilidade de replicar esses dados a outros participantes da transmissão de vídeo. O servidor possui uma conexão de sessão com cada um dos participantes, que permanece sempre ativa, e uma tabela contendo todos os participantes e o endereço IP de cada um deles. É o servidor que define quem vai receber o vídeo de quem e notifica os participantes enviando mensagens de controle.

No funcionamento do serviço CSMu, a comunicação de controle entre o servidor e todos os clientes é feita através de várias conexões de transporte sobre um serviço de sessão confirmado, o que caracteriza o conceito de multiponto. Na replicação dos dados entre os clientes participantes utiliza-se o conceito de conexões de transporte *multicast*, que pode ser

sobre um serviço de sessão confirmado ou não, dependendo da necessidade da aplicação. A Figura 15 apresenta a diferença de uma conexão de transporte multiponto e uma conexão de transporte *multicast*. No modo multiponto está representado o servidor, enquanto no modo *multicast* representa-se apenas a interação de um cliente que replica seu tráfego para os outros clientes participantes.

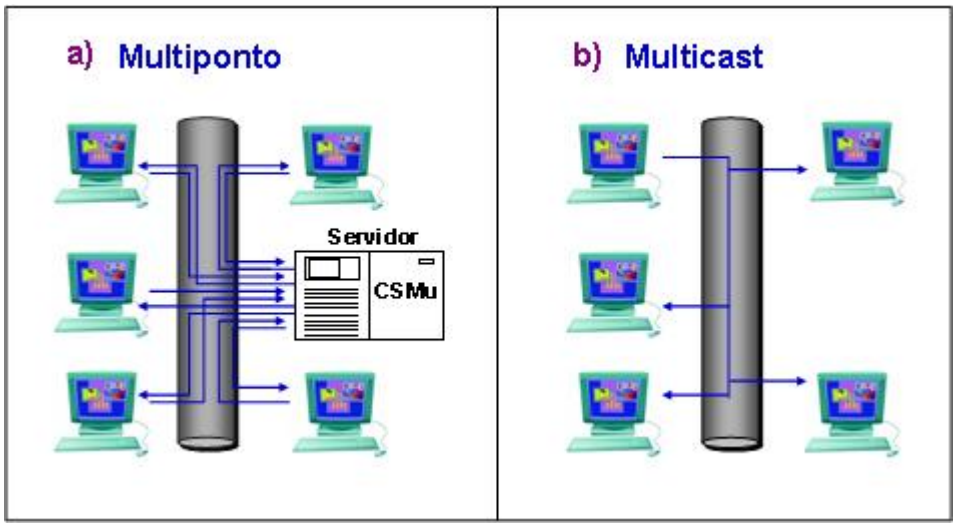


Figura 15 – Representação da conexão de sessão: a) multiponto (CSMu); b) *multicast*

O serviço CSMu é ideal apenas para o tráfego multimídia em tempo real, pois ele implementa um mecanismo distribuído de gerenciamento de buffer, onde dados mais antigos são descartados.

O serviço CSMu é tratado de forma diferente pela camada de sessão; nele um mesmo identificador de sessão é compartilhado entre todos os participantes. Isso reduz o nível de segurança, já que basta conhecer o id_session para poder participar da sessão. Porém, garantir a integridade dos dados não faz parte do escopo deste trabalho.

Esse tipo de serviço é ideal para conferências na Internet com vários participantes, podendo ser associado a um mecanismo de atividade, sinalizando o início de uma atividade, e a um mecanismo de token, criando a figura de um moderador (descritos no item 4.4). Por exemplo, em um evento do MBone, esses id_session podem ser divulgados na rede ou oferecidos aos participantes da conferência. Nesse caso, uma forma de fazer a divulgação é a aplicação servidora solicitar a camada de sessão a criação de um id_session CSMu com um tempo de expiração bem grande (que represente semana ou mês), e divulgar o número do id_session CSMu em uma de suas ferramentas.

O MBone utiliza a ferramenta SDR para anúncio de sessão, em que o cliente pode escolher a sessão em que ele quer participar. Atualmente, a divulgação dessas conferências

utiliza um endereço *multicast* IP classe D, quase sempre barrado nos roteadores e *firewalls* da rede. Esse problema pode ser solucionado em nível de camada de sessão utilizando-se o serviço CSMu.

A Figura 16 apresenta o autômato que descreve o estabelecimento de uma conexão CSMu. Mais detalhes podem ser vistos no Apêndice A.

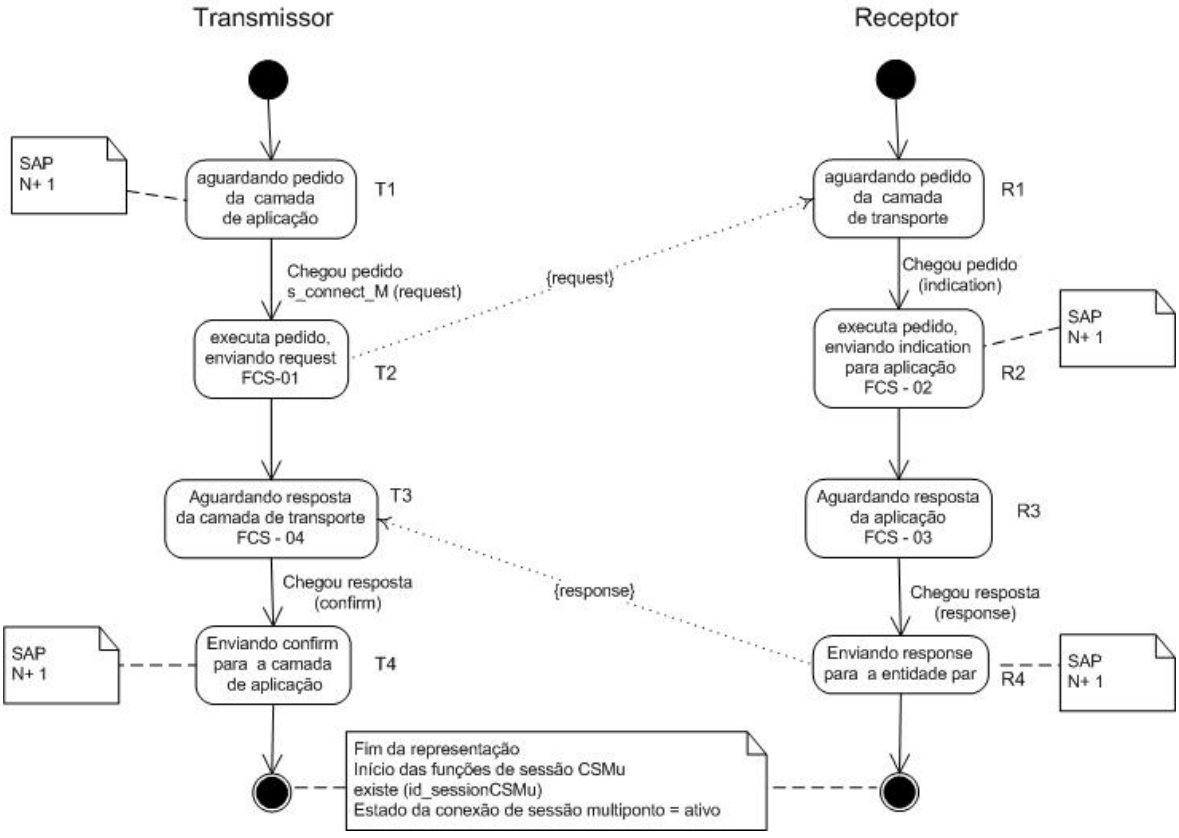


Figura 16 – Estabelecimento de uma sessão CSMu

4.4 Componente Funções de Sessão (FS)

O componente Funções de Sessão (FS) contempla as funcionalidades básicas de sessão. Este módulo só entra em atividade se existir um id_session e se os parâmetros da sessão tiverem sido negociados. Toda negociação e todos os serviços de controle utilizam um serviço de sessão confirmado; apenas o serviço de envio de dados pode ser executado no modo confirmado ou no modo não confirmado.

O componente FS oferece os serviços de transferência de dados, sincronização, controle de diálogo e gerência de atividade, que serão descritos, com mais detalhes, nos itens 4.4.1 a 4.4.4.

4.4.1 Transferência de Dados

Na extensão proposta, existem três tipos de dados que podem ser usados pelo usuário do serviço de sessão: dados normais, dados classificados e dados de capacidade. A aplicação pode escolher o tipo de transferência a ser utilizado (confirmado ou não confirmado), através das primitivas apresentadas no Quadro 9.

Os dados normais podem ser enviados por um serviço de sessão confirmado ou não confirmado, dependendo da necessidade da aplicação.

Dados classificados são os dados formatados para um determinado padrão, como, por exemplo, o padrão MIME. Estes dados também podem ser enviados por um serviço de sessão confirmado ou não confirmado.

Apensar de não se ter encontrado nenhuma aplicação multimídia da Internet que utilizasse dados urgentes. O padrão WSP também não suporta esse tipo de dado. Esse tipo de dado é mais utilizado por aplicações de gerenciamento da rede, quando existem problemas a serem solucionados, cujos pacotes de controle não podem estar sujeitos às filas de transmissão. Entretanto, dados urgentes podem vir a ser utilizados pelas aplicações multimídia para garantir a qualidade de serviço. Em redes de pacotes, um congestionamento pode atrapalhar a qualidade de uma transmissão de vídeo devido ao descarte dos pacotes nos roteadores. Neste caso, os dados urgentes podem ser utilizados para sinalizar os clientes envolvidos na transmissão.

Dados de capacidade são enviados apenas por um serviço de sessão confirmado (sobre TCP) e necessitam de confirmação também em nível de camada de sessão, por isso utilizam as quatro primitivas: *request, indication, response* e *confirm*. São utilizados no processo de estabelecimento de sessão onde são negociados os parâmetros de controle do protocolo de sessão: id_session, id_service, id_classes e id_categoria, definidos no Quadro 8 do ítem 4.3.1.

O Quadro 9 apresenta as primitivas utilizadas no serviço de transmissão de dados, e a Figura 17 apresenta o autômato. Mais detalhes estão descritos no Apêndice A.

PRIMITIVA	R	I	Rs	C	FUNÇÃO
Serviço com Sessão					
S_DATA_CONF	•	•			Serviço de transferência de dados normais confirmado (TCP).
S_DATA_NOT_CONF	•	•			Serviço de transferência de dados normais não confirmado (UDP).
S_TYPED_DATA_CONF	•	•			Serviço de transferência de dados classificados confirmado (TCP)
S_TYPED_DATA_NOT_CONF	•	•			Serviço de transferência de dados classificados não confirmado (UDP).
S_EXPEDITED_DATA_CONF	•	•			Serviço de transferência de dados urgentes confirmado (sobre TCP)
S_EXPEDITED_DATA_NOT_CONF	•	•			Serviço de transferência de dados urgentes não confirmado (UDP).
S_CAPABILITY_DATA	•	•	•	•	Serviço de transferência de dados de capacidade (apenas TCP).

Quadro 9 – Primitivas utilizadas no envio de dados

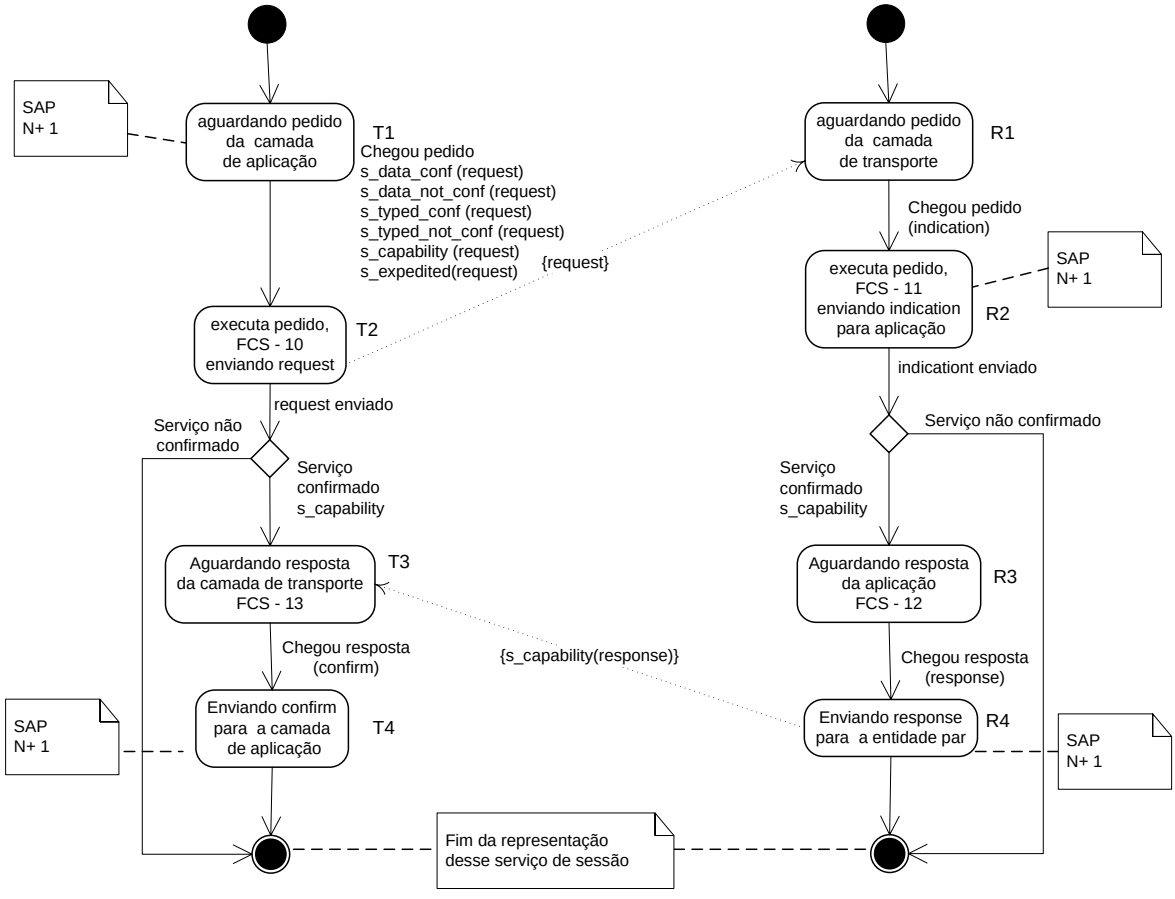


Figura 17 – Envio de dados

4.4.2 Sincronização e Ressincronização

O mecanismo de sincronização foi simplificado. Diferente do RM-OSI, existe apenas um tipo de ponto de sincronismo inserido na fronteira da informação. A inserção de um ponto de sincronismo deve ser explicitamente reconhecida pelo receptor. Uma vez que o servidor recebe a confirmação de que o cliente recebeu um novo ponto de sincronismo, os dados anteriores a esse ponto podem ser descartados.

Como visto anteriormente, no Capítulo 3, a transmissão de dados multimídia exige técnicas de bufferização. O controle da alimentação desses buffers é uma redundância presente na maioria das aplicações multimídia. O modelo proposto permite utilizar mecanismos de sincronização na alimentação desses buffers, implementando os protocolos RTP e RTCP dentro da camada de sessão, liberando, assim, a camada de aplicação desta tarefa.

As primitivas de sincronização que permitem a alimentação desse buffer são definidas no Quadro 10.

PRIMITIVA	R	I	Rs	C	FUNÇÃO
Serviço de Sessão Confirmado					
S_SYNC	•	•	•	•	Inserção de ponto de sincronização
S_RESYNCHRONIZE	•	•	•	•	Pedido de ressincronização

Quadro 10 – Primitivas utilizadas na sincronização

Os mecanismos de sincronização do modelo implementam apenas um ponto de sincronismo. Para cada pedido de ressincronização recebido, o buffer de sincronização retorna apenas um estágio para trás (que corresponde ao último ponto de sincronismo confirmado). A máquina de estados do mecanismo de sincronização pode ser vista na Figura 18. Mais detalhes do funcionamento da máquina bem como os algoritmos de controle desses buffers são apresentados no Apêndice A.

Para garantir o funcionamento do serviço de sincronização, é preciso existir pelo menos um ponto de sincronismo. Caso, futuramente, exista a necessidade de mais de um ponto de sincronismo, visando obter maior granularidade, este recurso pode ser implementado no componente Extensões Futuras.

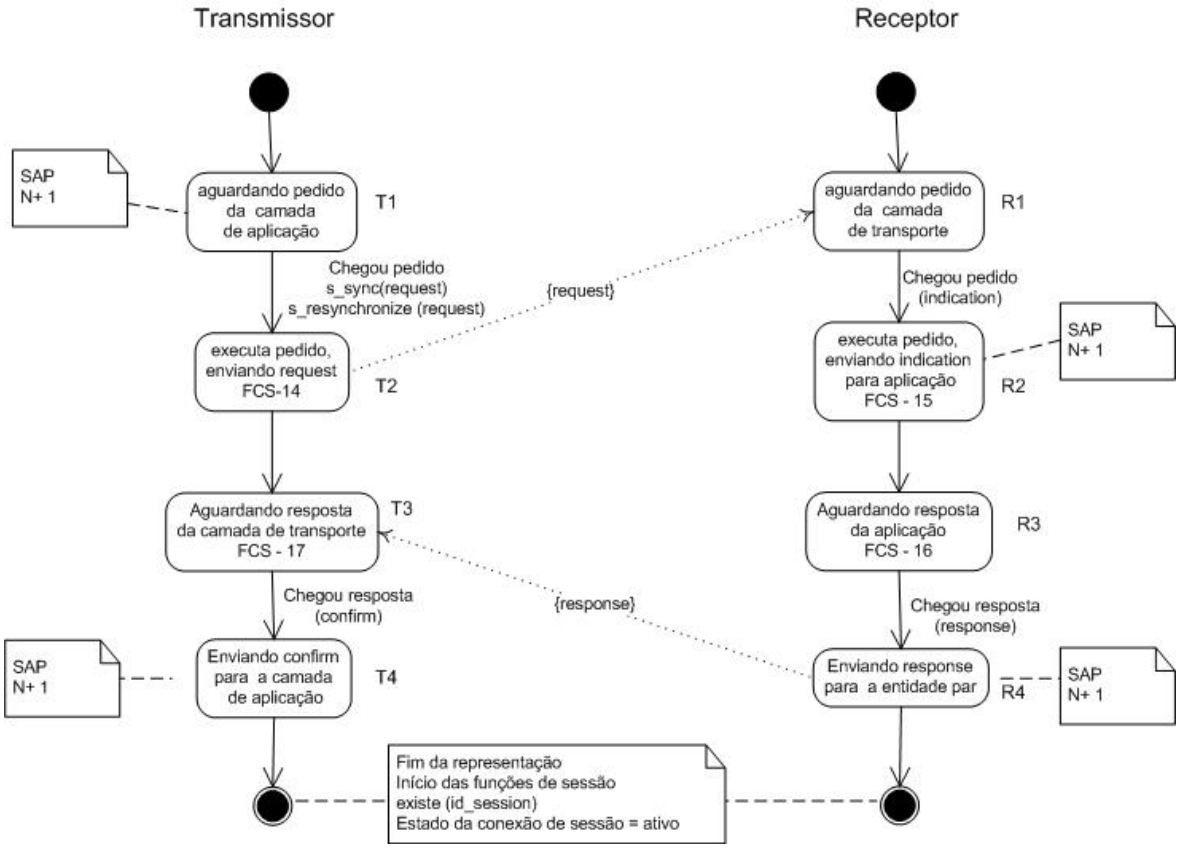


Figura 18 – Serviço de sincronização

4.4.3 Gerência de Diálogo

Como descrito no Apêndice B, gerência de diálogo é o serviço de sessão que permite controlar o diálogo entre entidades participantes. Utiliza um mecanismo de passagem de fixa[15], onde o detentor da fixa faz uso de certos serviços com exclusividade.

Na extensão proposta, o serviço de controle de diálogo é proposto de uma forma simplificada, funcionando apenas no modo *half-duplex*, onde o servidor (que gerou o id_session) é o responsável pelo controle de posse da palavra.

Quando uma aplicação requer que a transferência de dados seja feita na forma *half-duplex*, ela deve negociar a utilização da gerência de diálogo durante o estabelecimento da conexão, através dos parâmetros de serviço. A ficha que habilita a transferência de dados pode ser ou não ser enviado ao usuário, dependendo da "vontade" do moderador (servidor). Esse

[15] Token (ou fixa de dados) pode ser implementado através de uma FLAG, que, quando verdadeira, sinaliza que a entidade de sessão é detentora do token e está habilitada para transmitir dados.

mecanismo de gerência de diálogo pode ser utilizado, por exemplo, em aplicações multimídia, tais como aplicações de videoconferência com controle de posse da palavra.

> O controle de posse de palavra aplica-se aos canais de áudio e vídeo mas, de forma mais abstrata, seu conceito e funcionamento podem ser aplicados a qualquer outro recurso síncrono e interativo que precise de acesso coordenado, resolvendo conflitos no seu uso, como condições de corrida e *deadlocks*. (SMETANA, 2004).

As primitivas de serviço utilizadas na gerência de diálogo da camada de sessão estão descritas no Quadro 11 e a sua concepção é bastante similar ao RM-OSI. Entretanto, para implementar a funcionalidade de um moderador, o pedido de ficha é confirmado. Em nível de camada de sessão, a entidade que gerou o id_session atuará como moderadora, atendendo ao pedido das demais entidades.

PRIMITIVA	R	I	Rs	C	FUNÇÃO
Serviço Com Sessão Confirmado					
S_TOKEN_GIVE	•	•			Passagem de ficha de dados
S_TOKEN_PLEASE	•	•	•	•	Pedido de ficha
S_CONTROL_GIVE	•	•			Passagem de todas as fichas

Quadro 11 – Primitivas de gerência de diálogo

O autômato da Figura 19 detalha o funcionamento do mecanismo de gerência de diálogo entre dois transmissores (clientes) e um receptor (servidor), numa conexão CSMu. O transmissor é quem inicia a sessão. Da mesma forma que o id_session, a ficha é gerada pelo receptor; logo, ele é o moderador. Mais detalhes desse serviço são descritos no Apêndice A.

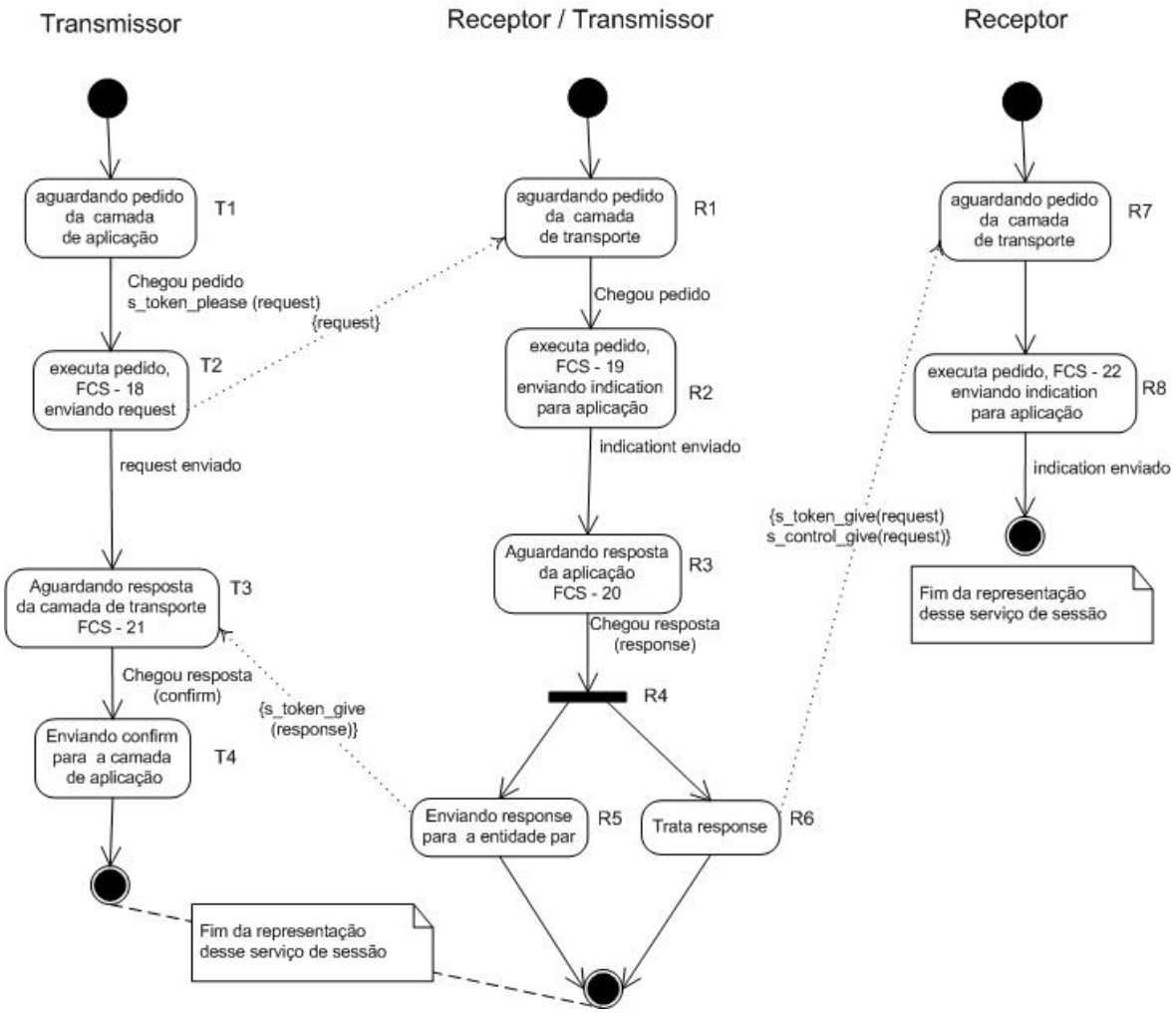


Figura 19 – Gerência de diálogo

4.4.4 Gerência de Atividades

Para a funcionalidade de gerência de atividades não foi feita nenhuma consideração diferente da especificada pelo RM-OSI, dado que a funcionalidade de gerência de atividade não foi identificada em nenhuma das aplicações multimídia avaliadas. Entretanto, sabe-se que aplicações bancárias e de comércio eletrônico utilizam o conceito de atividades. Nesse sentido, manteve-se na extensão o conceito definido pelo modelo RM-OSI, para que ela possa ser utilizada no desenvolvimento de aplicações que necessitem dessa funcionalidade de sessão.

De acordo com o conceito de atividades definido pelo RM-OSI, uma sessão completa pode conter várias atividades consecutivas; porém, num dado instante, somente uma atividade

pode estar em progresso em uma conexão de sessão. Nesse sentido, sempre existe apenas uma atividade associada a um id_session.

O Quadro 12 descreve as primitivas utilizadas para controle de atividade, e a Figura 20 apresenta o autômato.

PRIMITIVA	R	I	Rs	C	FUNÇÃO
Serviço Com Sessão Confirmado					
S_ACTIVITY_START	•	•			Início de uma atividade
S_ACTIVITY_END	•	•	•	•	Fim de uma atividade
S_ACTIVITY_DISCARD	•	•	•	•	Abandono de uma atividade
S_ACTIVITY_INTERRUPT	•	•	•	•	Interrupção de uma atividade
S_ACTIVITY_RESUME	•	•			Retomada de uma atividade

Quadro 12 – Primitivas de atividade

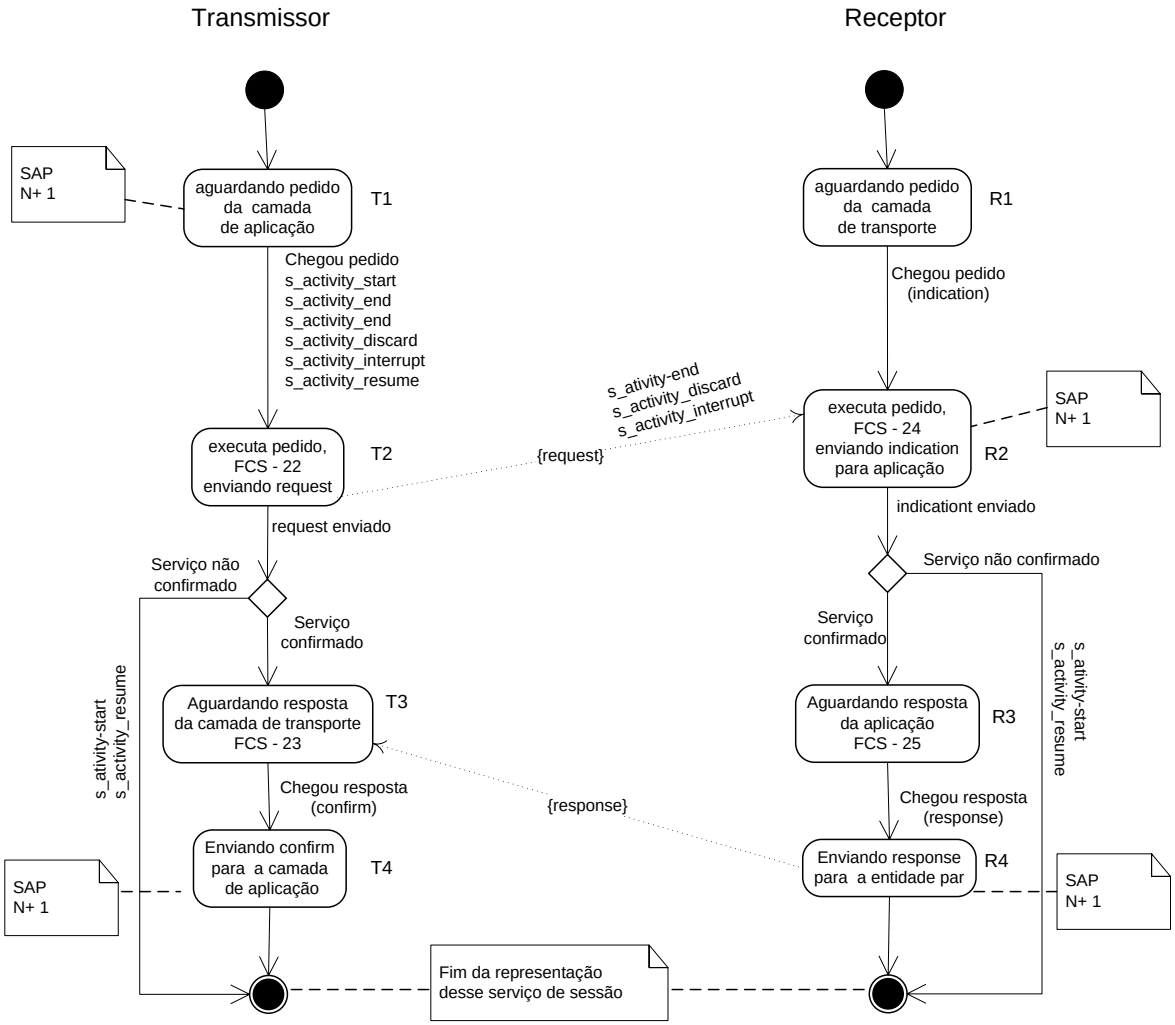


Figura 20 – Autômato de atividade

4.5 Componente Gerência de Buffer (GB)

Como visto anteriormente, no Capítulo 3, aplicações de tempo real requerem técnicas de bufferização, que atualmente são implementadas dentro de seu próprio código. Um dos objetivos do componente Gerência de Buffer é oferecer espaço para a programação desses algoritmos, de forma que possam ser utilizados por todas as aplicações, sem que eles tenham a necessidade de ser inseridos dentro do código de cada aplicação, o que ameniza o problema de redundância de código. Dessa forma, protocolos que compõem a infra-estrutura multimídia e novos algoritmos de controle de buffer podem ser inseridos na camada de sessão, sendo seus serviços oferecidos à camada de aplicação. Por ser um componente separado, o GB permite a adição de novos algoritmos sem prejudicar o funcionamento dos demais componentes.

Outra funcionalidade oferecida no componente GB é o serviço de player. Este serviço possibilita que aplicações multimídia utilizem funções de avanço e retrocesso, quando estão buscando dados em um buffer local. Como visto na seção 3.3.3, o protocolo RTSP permite que um cliente execute funções de avanço e retrocesso em um servidor remoto de fluxo de vídeo. Entretanto, com o fluxo de dados centralizado no servidor, tem-se um canal de transmissão de dados aberto para cada um dos clientes, o que gera uma sobrecarga no servidor e na rede, como explicado na seção 3.5.

O GB utiliza dois tipos de buffers: um de sincronização e outro de player-out. O buffer de sincronização é alimentado pela função de sincronização no processo de transferência de dados, como descrito no item 4.4.2. O buffer de player é um serviço de buffer local, que possui mecanismos de avanço e retrocesso. Um mecanismo de controle local descentraliza do servidor de *streaming* remoto as funções de avanço e retrocesso, permitindo assim reduzir o *overhead* gerado nos servidores de *streaming*.

O buffer de player funciona de duas maneiras: apenas oferecendo serviços para a aplicação local ou compartilhando seus dados entre vários clientes distribuídos. No último caso, é necessário que uma conexão multiponto (CSMu) tenha sido estabelecida previamente.

4.5.1 A Função de Player Local

A função de player é direcionada para aplicações multimídia de vídeo sob demanda (VoD). Assim como nas demais funcionalidades da camada de sessão, para que os serviços possam ser oferecidos, um id_session precisa existir.

O mecanismo de player armazena em um buffer local todos os dados enviados do servidor. Quando funcionalidades de avanço e retrocesso são solicitadas, a consulta é realizada, primeiramente, no buffer local. Dessa forma, o dado só será requisitado à entidade remota se ele não estiver armazenado no buffer local (player), nem no buffer de sincronização. Essa técnica reduz o uso da rede, já que funções de avanço e retrocesso associadas a uma determinada aplicação são tratadas localmente, por meio de um serviço oferecido entre camadas, e não entre um cliente e um servidor remoto. Esse processo pode ser visualizado na Figura 21.

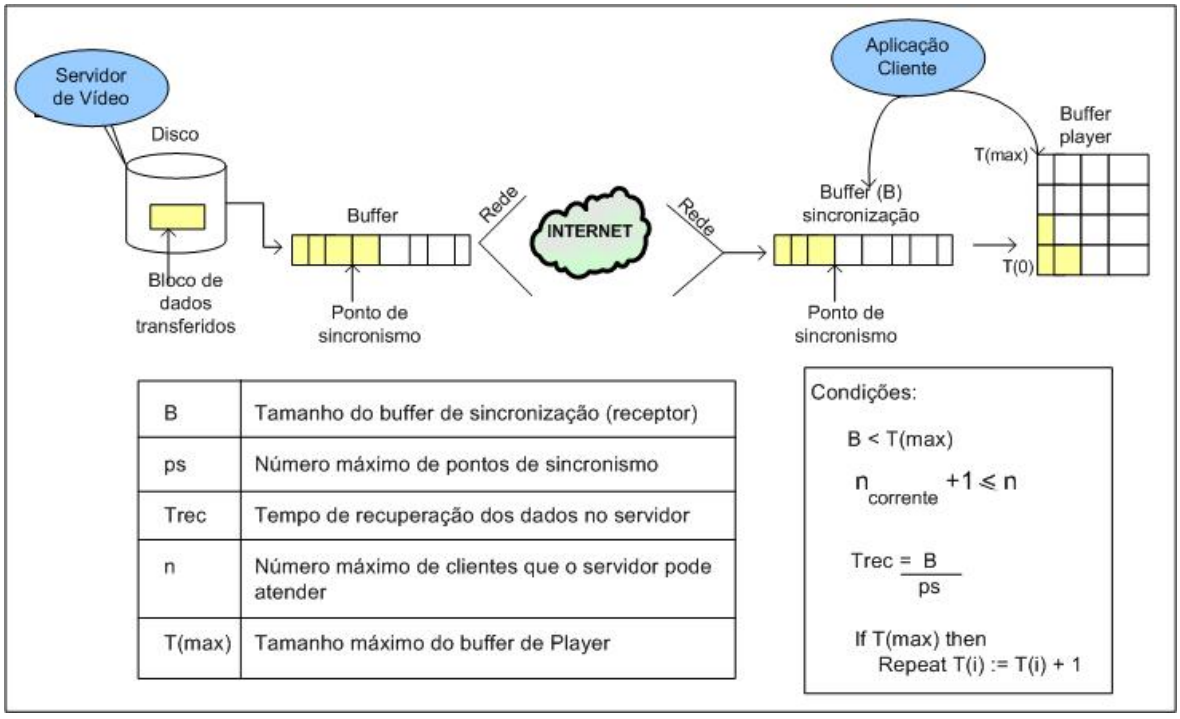


Figura 21 – Funcionamento do buffer de player

Sendo B o tamanho do buffer de sincronização do lado receptor e T o tamanho do buffer de player, B sempre deve ser menor que T. Como cada aplicação multimídia possui seus requisitos específicos de atraso, o tamanho dos buffers pode ser especificado pela aplicação e é ela que define os pontos de sincronismo através da utilização das primitivas de serviços de sincronização (s_sync) do componente FS. O parâmetro n, que indica número máximo de clientes, também é definido pela aplicação. A taxa de recuperação de dados do servidor é igual ao tamanho do buffer B dividido pelo número máximo de pontos de sincronismo (marcados pela aplicação).

Quando o buffer de sincronização estiver cheio, antes de os dados serem descartados, eles são migrados para o buffer de player-out. O método FIFO, descrito no item 3.5, deve ser utilizado na programação dessa funcionalidade. Sempre que o buffer de player-out (T) estiver cheio (max), ocorre um deslocamento, quando os dados mais antigos são descartados.

As primitivas de serviço, descritas no Quadro 13, são utilizadas na implementação do buffer de player local, cuja máquina de estados está descrita na Figura 22. Mais detalhes de implementação podem ser obtidos no Apêndice A.

PRIMITIVA	R	I	Rs	C	FUNÇÃO
Serviço com Sessão Confirmado					
S_PLAYER_START	•				Início do mecanismo de player
S_PLAYER_END	•				Fim do mecanismo de player
S_PLAYER_PLAY	•				Aplicação sinaliza que está acessando os dados do buffer de player
S_PLAYER _PAUSE	•				Aplicação sinaliza que parou de acessar os dados do buffer de player
S_PLAYER_UP	•				Função de avanço
S_PLAYER_DOWN	•				Função de retrocesso

Quadro 13 – Primitivas do buffer de player

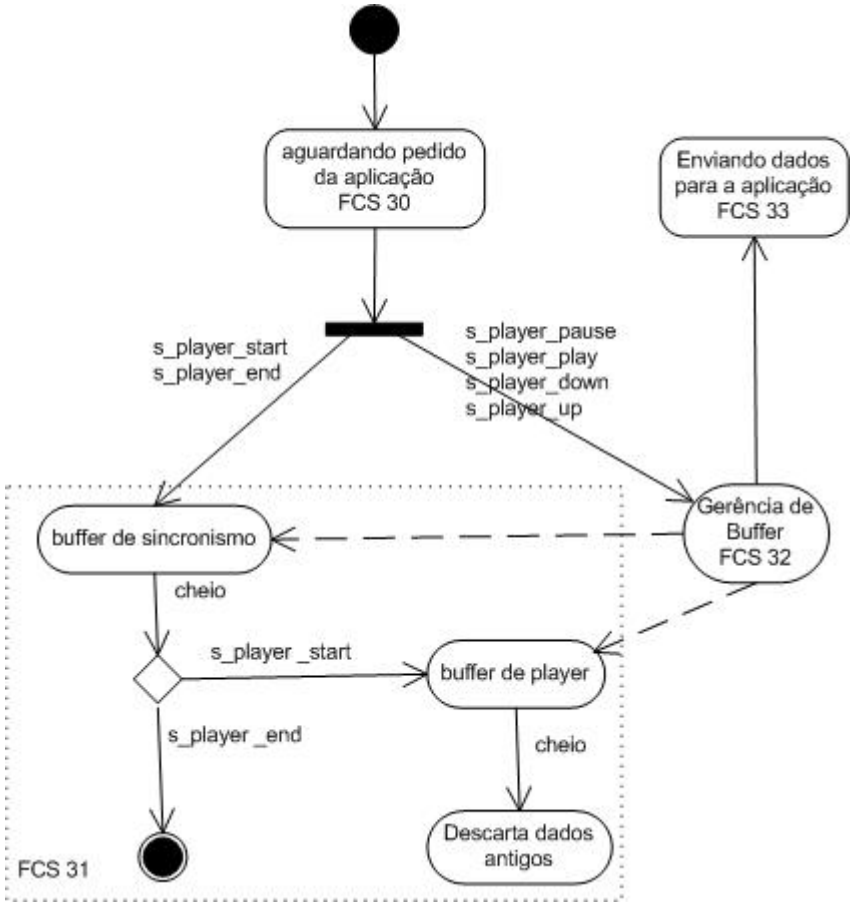


Figura 22 – Funcionamento do buffer de player

Quando a aplicação inicializa um mecanismo de player, através da primitiva s_player_start, ela passa a consumir os dados do buffer de player-out. Quando a aplicação insere um ponto de sincronização, ela interage com o buffer de sincronização. O componente GB deve efetuar a cópia dos dados do buffer de sincronização para o buffer de player-out. Através das primitivas de serviço (s_player_play, s_player_pause, s_player_up, s_player_down) a aplicação comanda a apresentação dos dados. Dados mais antigos são descartados da memória virtual, podendo ou não ser armazenados em disco. Ao acionar a primitiva s_player_end, a aplicação passa a consumir os dados diretamente do buffer de sincronização, não existindo mais o recurso de player.

4.5.2 A Função de Player Distribuído

De acordo com Sen e Towsley (1999), é possível minimizar a sobrecarga de um servidor de vídeo mediante a utilização de servidores *proxy*. Porém, esses servidores precisam ser instalados e configurados em locais dispersos na rede. A função de player distribuído dispensa

a configuração de servidores *proxy*, pois a própria camada de sessão implementa um mecanismo de gerência de buffer distribuído capaz de otimizar a utilização da largura de banda da rede.

A função de player distribuído depende da existência de uma CSMu. Quando uma aplicação faz um pedido de sessão multiponto a um servidor remoto (utilizando uma primitiva s_connect_m), ela informa ao servidor que os dados de seu buffer de player podem ser disponibilizados a outros participantes. Quando o servidor recebe uma primitiva s_player_indication, ele sabe que o buffer de player do cliente está em atividade. A partir deste ponto, o servidor pode indicar a outros participantes a possibilidade de acessar diretamente este cliente, através de uma sessão CSMu. Esta funcionalidade pode ser visualizada na Figura 23.

Utilizar o serviço de buffer de player distribuído permite diminuir a carga no servidor, pois distribui o fluxo de dados entre todos os participantes da sessão. O primeiro cliente a se conectar recebe os dados do servidor, o segundo cliente recebe os dados do primeiro, e o terceiro cliente, do segundo. A largura de banda da rede é mais bem utilizada, pois cada receptor (cliente) pode também funcionar como emissor (servidor) de tráfego multimídia, o que configura uma funcionalidade típica de redes P2P (*peer-to-peer*).

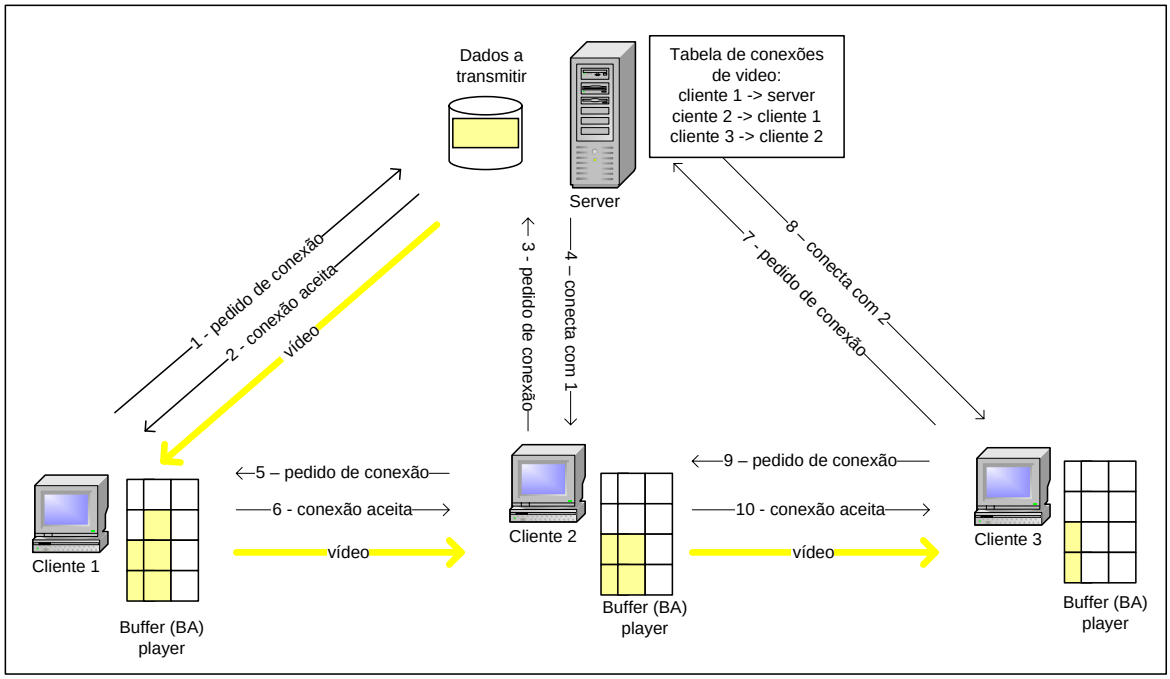


Figura 23 – Funcionamento do controle de sessão multiponto

O servidor que gerou o identificador de sessão multiponto (CSMu) é responsável por implementar a gerência do buffer. Ele utiliza conexões de transporte multiponto com os participantes por onde passam mensagens de controle. Possui uma tabela com informações sobre cada um dos participantes, utiliza o endereço de rede para calcular o menor caminho e controla o número máximo de clientes que cada participante pode atender.

Entre os participantes implementa-se um mecanismo de comunicação *multicast*. Cada cliente pode replicar os dados de seu buffer de player a outros participantes, através de um ou mais canais de transporte associados a uma mesma conexão de sessão (CSMu). Um caso de uso desta funcionalidade é apresentado no Apêndice A (caso de uso 10).

Observando-se a Figura 23, o cliente 2 recebe dados do cliente 1. Como o fluxo de dados que ele recebe não vem diretamente do servidor, pode ocorrer que os dados mais antigos não estejam mais armazenados no buffer do cliente 1, pois o espaço do buffer precisou ser liberado para receber dados mais recentes. Por esse motivo, a função de player distribuído é destinada exclusivamente às aplicações multimídia de tempo real.

As primitivas de serviço utilizadas no serviço de player distribuído estão descritas no Quadro 14. A Figura 24 apresenta os autômatos que definem o funcionamento deste componente.

PRIMITIVA	R	I	Rs	C	FUNÇÃO
Serviço com Sessão Confirmado					
S_PLAYER_PUSH_IN	•	•			Pedido de envio dos dados do buffer de player
S_PLAYER_PUSH_OFF	•	•			Envio dos dados do buffer

Quadro 14 – Primitivas do buffer de player distribuído

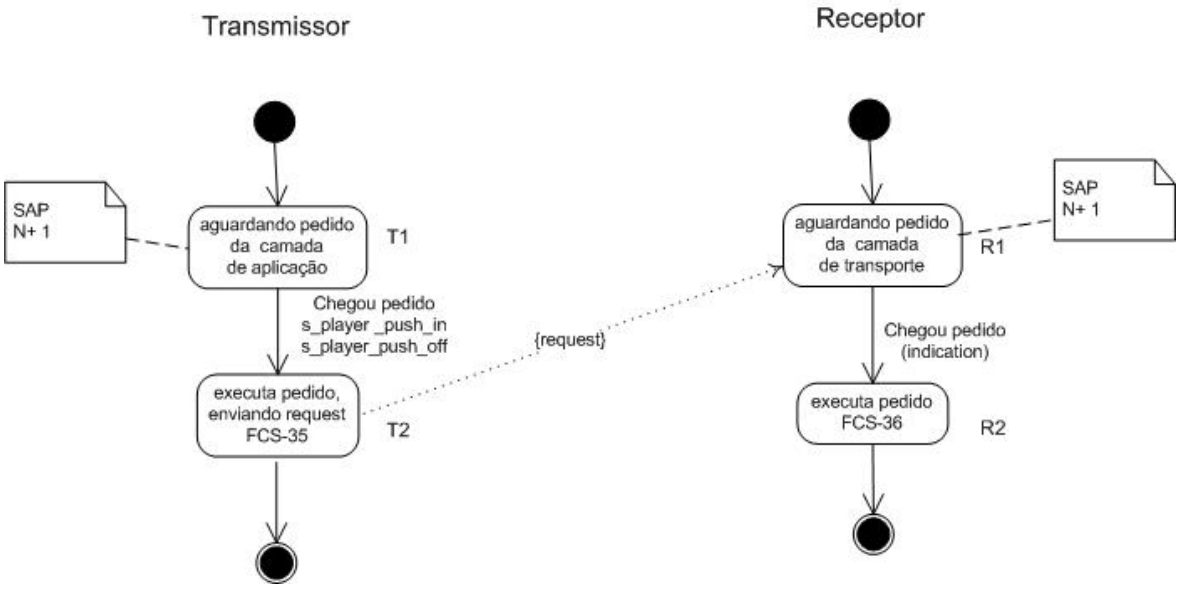


Figura 24 – Autômato do buffer de player distribuído

4.6 Componente Análise de Tráfego (AT)

Como abordado no Capítulo 3 (item 3.2.2), as propostas de análise de tráfego existentes utilizam as portas padrão, definidas pelo IANA, para identificar o tipo de tráfego. Para o tráfego elástico a associação é direta, porém para o tráfego inelástico esse recurso é insuficiente, pois aplicações multimídia alocam portas dinâmicas na transmissão de dados. No exemplo da solução proposta no *mmdump* (MERWE, 2000), um *parser* foi construído associando-se uma sessão a todas as portas de transporte abertas durante o funcionamento de uma aplicação multimídia. Os autores mencionam a necessidade de replicar o *parser* (construído para interpretar as portas do protocolo RTSP e H.323) para cada um dos protocolos multimídia existentes. A extensão proposta permite inserir uma camada de sessão na arquitetura da Internet e, dessa forma, o id_session pode ser utilizado para mapear o tráfego inelástico, excluindo, assim, a necessidade de ter-se um mecanismo de *parser* específico para cada um dos protocolos multimídia. Dessa forma, propõe-se criar um componente para análise de tráfego dentro da camada de sessão.

O componente Análise de Tráfego (AT) permite separar e agrupar o tráfego em categorias, oferecendo serviços de categorização e contabilização para a camada de aplicação (necessidade abordada no Capítulo 3). Existem duas formas de categorizar o tráfego. A primeira divide o tráfego em apenas duas categorias: elástico e inelástico. A segunda permite uma categorização mais detalhada, mas depende do preenchimento do campo id_categoria (visualizado no Quadro 8), realizado pelas aplicações durante o processo de estabelecimento de uma conexão de sessão.

A definição de duas abordagens baseou-se no funcionamento atual das aplicações da rede, mantido pela extensão através dos serviços de *pass-throught*; e em uma abordagem futura, considerando as aplicações que utilizarão os serviços propostos nessa extensão.

O componente AT foi separado dos demais porque seu funcionamento não exige o estabelecimento de uma sessão entre duas entidades pares. Ele funciona como um coletor de dados da camada de sessão, que oferece serviços de contabilização de tráfego apenas para a entidade local.

O componente AT permite detectar o tipo de tráfego que passa pela camada de sessão local. Toda a informação sobre o tráfego que passa pela camada de sessão é armazenada em uma área indicada pela aplicação. A camada de sessão alimenta esta área com os dados

coletados, e a aplicação pode ler essas informações. Esta funcionalidade pode ser visualizada na Figura 25 e está descrita em detalhes no item 4.6.1.

Esse serviço é disponibilizado para a aplicação através das primitivas do Quadro 15.

PRIMITIVA	R	I	Rs	C	FUNÇÃO
S_TRAF_START S_TRAF_END	• •				Início da coleta de tráfego Fim da coleta de tráfego

Quadro 15 – Primitivas do componente AT

4.6.1 Funcionamento do Componente AT

Como mencionado anteriormente, quando uma aplicação deseja obter informações sobre o tráfego, ela cria uma área de armazenamento e aciona a primitiva s_traf_start da camada de sessão. Quando a aplicação deseja encerrar a coleta de tráfego, ela o faz através da primitiva s_traf_end. Se a área de armazenamento definida pela aplicação estiver cheia, a camada de sessão suspende a contabilização do tráfego. A aplicação lê os dados coletados e trabalha-os da forma que desejar, seja gerando estatísticas numéricas sobre esses dados, seja gerando ambientes gráficos. A Figura 25 esboça o funcionamento do componente AT.

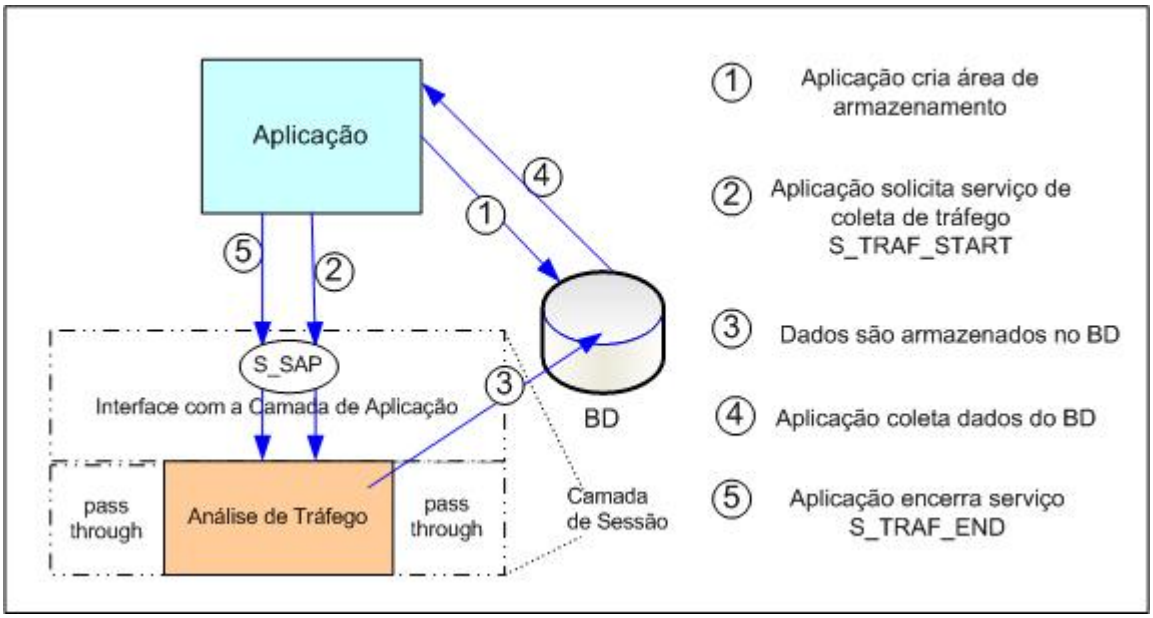


Figura 25 – Funcionamento do componente AT

4.6.2 Divisão do Tráfego em Elástico e Inelástico

O modelo estabelece duas formas de categorizar o tráfego dentro da camada de sessão, e uma delas utiliza o serviço de *pass-through*. No serviço de *pass-through* a camada de sessão apenas monitora as conexões de transporte abertas por uma determinada aplicação. Para fins de contabilização de tráfego, a instância da aplicação e as informações sobre os canais de comunicação abertos por ela (portas TCP/UDP utilizadas) são armazenadas no registro da camada de sessão, cuja análise permite dividir o tráfego em duas categorias: elástico e inelástico.

Nesta primeira forma de categorização de tráfego, as duas premissas abaixo são consideradas válidas:

a) o campo id_categoria possui um valor nulo ou não válido. Isso ocorre quando a aplicação não preenche o campo id_categoria da PDU utilizada no processo de estabelecimento de conexão de sessão; e
b) protocolos multimídia utilizam portas conhecidas padronizadas pelo IANA.

Atendendo as premissas acima, além do serviço de *pass-through*, o tráfego que passa pela camada de sessão também pode utilizar um serviço da própria camada de sessão. Em ambos os casos as seguintes informações são utilizadas para categorizar o tráfego:

a) a porta padrão IANA utilizada pela aplicação para o protocolo de transporte;
b) um identificador da aplicação (por exemplo, PID no Unix); e
c) a classe de serviço de sessão negociado (classes dos componentes FS, GB) em tempo de estabelecimento de sessão.

O Quadro 16 apresenta um exemplo da aplicação do método de categorização de tráfego adotado nesta primeira abordagem.

Aplicação	**Classes de Serviço de Sessão**	**Porta**	**Tráfego**
Tomcat	Sessão confirmada (estabelecimento de sessão, sincronização)	8080	Elástico
Apache	*Pass-through*	80	Elástico
Q-mail	*Pass-through*	25	Elástico
WebConference	Sessão confirmada e não confirmada (estabelecimento de sessão, função de player, sincronização)	Várias	Inelástico
VideoLAN	*Pass-through*	554	Inelástico

Quadro 16 – Categorização do tráfego entre elástico e inelástico

Para entender o funcionamento do modelo, considera-se como exemplo uma aplicação multimídia que utilize o protocolo RTSP. Neste caso, quando a aplicação acessa um serviço de *pass-through*, a camada de sessão intercepta o pedido e verifica em qual porta de transporte a aplicação deseja abrir um socket (veja autômato da Figura 26). Se o socket for aberto na porta 554 ou 8554 (TCP ou UDP), de acordo com o IANA, o protocolo utilizado é o RTSP, logo a aplicação gera tráfego inelástico. A camada de sessão identifica a instância da aplicação, e todos os outros pedidos de conexão de transporte dessa aplicação (que utilizam portas aleatórias) também serão mapeados como tráfego inelástico. O mesmo ocorre para os demais protocolos que compõem as aplicações multimídia e possuem portas padronizadas pelo IANA, tais como RTP, RTCP e outros.

Esta alternativa permite avaliar o tráfego das aplicações existentes que desconhecem a existência da camada de sessão e utilizam o mecanismo de *pass-through*. A técnica adotada utiliza como base o exemplo do *mmdump*, descrito na seção 3.2.2. Entretanto, nesta abordagem as sessões são mapeadas antes da abertura dos sockets da camada de transporte, dispensando a utilização de mecanismos de *parser*[16].

Como explicado anteriormente, os mecanismos de player e player distribuído, implementados no componente GB, são destinados às aplicações multimídia. Logo, sempre que esses serviços forem solicitados por uma aplicação, no momento de estabelecimento de sessão, todo o tráfego relacionado ao id_session dessa aplicação será mapeado como inelástico.

4.6.3 Divisão do Tráfego em Várias Categorias

A segunda forma, proposta neste modelo, de qualificar o tráfego da Internet utiliza a própria estrutura da camada de sessão. Essa qualificação leva em consideração o preenchimento obrigatório, pela aplicação, do parâmetro id_categoria e o facultativo do parâmetro id_instância, ambos utilizados na PDU de negociação do serviço de estabelecimento de sessão. O campo id_categoria possui valores que permitem identificar mais detalhamente o tráfego da aplicação. Exemplos desses valores estão descritos no Quadro 17.

[16] Mecanismos de *parser* são utilizados pelo *mmdump* para identificar a aplicação e todas as conexões de transporte abertas por ela.

ID_CATEGORIA	DESCRIÇÃO
1	Categoria Web
2	Categoria de correio eletrônico
3	Categoria de transferência de arquivo
4	Categoria de conexão remota
5	Categoria de serviços de rede
6	Categoria de serviço de voz sobre IP
7	Categoria de tráfego multimídia em tempo real (ex: conferência)
8	Categoria de vídeo sob demanda (ex: televisão *online*)
9	Categoria de áudio sob demanda (ex: rádio)
10	Categoria de dispositivo móvel

Quadro 17 – Tipos de categorias de tráfego

No momento de estabelecimento da conexão de sessão, a aplicação deve identificar o tipo de tráfego que ela pretende utilizar. Isso é feito mediante o preenchimento do campo id_categoria (Quadro 8), da PDU de conexão. Considerando que a primitiva s_traf_start foi acionada, nesta abordagem a camada de sessão verifica a informação do campo id_categoria, e todo o tráfego associado a esse id_session é mapeado de acordo com a categoria identificada.

O servidor remoto que gerou esse id_session associou a ele um determinado tipo de tráfego. Caso a aplicação tente utilizar o mesmo id_session para outro tipo de tráfego não negociado no momento de estabelecimento da sessão, o servidor gera um erro e aborta a sessão.

Além da divisão em categorias, a extensão proposta também permite a separação do tráfego em instâncias. Instâncias são elementos de uma determinada categoria. Por exemplo, para a categoria Web pode-se ter as seguintes instâncias: Tomcat, Apache, iPlanet, IIS (*Internet Information Services*).

Assim como existem as portas padrão do IANA, também podem existir padronizações para os diversos tipos de tráfego. Uma vantagem desta abordagem é a facilidade na identificação do tráfego e o alto nível de detalhamento, uma necessidade atual que pode aumentar com a utilização da rede. Porém, esta abordagem exige que as aplicações futuras sinalizem a categoria de tráfego que elas pretendem gerar.

4.7 Interoperabilidade

O serviço de *pass-through* garante a interoperabilidade da extensão proposta com as aplicações existentes. A camada de sessão intercepta a requisição da aplicação. Se a interface do serviço de sessão não foi utilizada, significa que a aplicação deseja abrir um socket TCP ou

UDP, seguindo o funcionamento tradicional da pilha da Internet. A camada de sessão abre o socket e passa o controle para a camada de aplicação, não interferindo em mais nada.

Essa interoperabilidade, oferecida pelo serviço de *pass-through*, é necessária para garantir a viabilidade do modelo diante das aplicações existentes. O autômato da Figura 26 demonstra o funcionamento do mecanismo de *pass-through*.

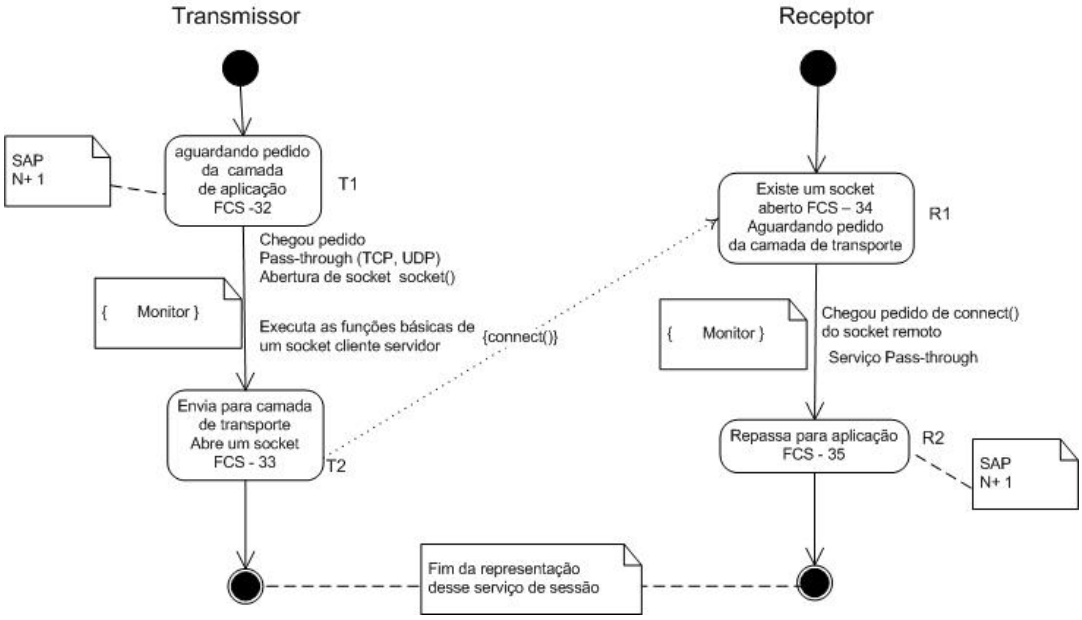


Figura 26 – Mecanismo de *pass-through*

4.8 Extensões Futuras

O componente Extensões Futuras foi criado para permitir que funcionalidades de sessão não previstas possam ser adaptadas ao modelo. Ele permite definir novas interfaces fazendo com que extensões futuras possam ser acrescentadas ao modelo sem necessidade de alterar os componentes existentes.

Extensões podem ser anexadas ao modelo aproveitando os serviços existentes e criando novas funcionalidades. Por exemplo, devido às pequenas dimensões dos dispositivos portáteis, pode ser interessante implementar o modelo proposto em hardware, e neste caso, as adaptações do modelo que têm ligação com os serviços de sessão, tais como tempo de inatividade da bateria do dispositivo portátil ou troca da área de *roaming*, são parte do componente Extensões Futuras. Dessa forma, projetistas de equipamentos podem usar o modelo tendo liberdade de inserir novas funções sem modificar as funções já definidas.

4.9 Considerações de Implementação

Inicialmente, avaliaram-se duas abordagens de implementação do modelo proposto, no kernel do sistema operacional ou como biblioteca de sistema. Uma proposta de protocolo de sessão (SESP) considerou as duas possibilidades (HENCHEN, 2004), em que os serviços de sessão foram implementados de forma análoga aos serviços definidos no modelo de referência RM-OSI e foram oferecidos às entidades de aplicações por meio da socket API. Utilizou-se o protocolo TCP para garantir a entrega das PDUs de sessão. Foi feita uma análise detalhada do código do Net/3 procurando todas as rotinas da camada de socket utilizadas pelo protocolo TCP; além de localizar as chamadas, foi alterado o código dos protocolos de transporte para que eles enviassem os dados vindos de sua interface de entrada ao protocolo de sessão, em vez de encaminhar aos buffers da camada de socket. A utilização do TCP como protocolo de transporte dispensou efetuar confirmação no envio de cada PDU de sessão. A recepção dos dados na arquitetura TCP/IP é coordenada por requisições da aplicação, por isso foi necessário que o protocolo de sessão possuísse rotinas que permitam a recepção de mensagens de controle de diálogo, sincronização e atividade. A maior dificuldade encontrada na implementação do SESP como núcleo do sistema operacional foi reorientar o fluxo e o tratamento de dados entrantes no sistema, ou seja, permitir que os dados que se destinam ao sistema passem pelo protocolo de sessão antes de chegar à aplicação destino. Além disso, como os dados das entidades de sessão são somente mantidos nos blocos de controle do protocolo, estas entidades podem ter seus dados perdidos no caso de uma falha do sistema operacional. Para poder garantir a persistência das entidades de sessão, a implementação deve garantir o armazenamento periódico em memória secundária do estado de cada entidade. Entretanto, acessos freqüentes ao disco rígido podem acarretar perda de desempenho, gerando um atraso no transporte de dados das aplicações que utilizarem o SESP. Se o tempo de gravação do estado da sessão em disco for muito grande, no pior dos casos, pode ocorrer inconsistência de dados entre as entidades de sessão comunicantes. Técnicas de consistência e log utilizadas em sistemas de gerência de banco de dados podem ser aplicadas na implementação para garantir a persistência das entidades de sessão.

Outra consideração relacionada à implementação refere-se à área utilizada pelos buffers, cujo tamanho e espaço de armazenamento podem ser definidos pela aplicação e repassados como parâmetro à camada de sessão. Entretanto, um projeto dessa extensão adaptada a dispositivos portáteis pode exigir a necessidade de fixar o tamanho dos buffers utilizados

dentro da camada de sessão. Detalhes podem sofrer adaptações de engenheiros de projetos de hardware, sendo abstraídos do contexto desta tese. Assim, deixa-se a decisão sobre a forma de implementação da extensão proposta como uma sugestão para trabalhos futuros.

4.10 Considerações Finais

A extensão proposta apresenta funcionalidades básicas que são essenciais na existência de uma camada de sessão, sendo obrigatório seu funcionamento entre duas entidades pares. Funcionalidades semelhantes aos serviços de sessão definidos no modelo RM-OSI são oferecidas em dois dos cinco componentes da camada de sessão: Controle de Sessão e Funções de Sessão.

Adaptadas ao contexto da Internet têm-se as seguintes funcionalidades básicas:

a) estabelecimento e liberação de conexão;
b) relatório de anomalias;
c) transferência de dados (normais, classificados e de capacidade);
d) sincronização;
e) gerência de diálogo; e
f) gerência de atividades.

A pesquisa bibliográfica realizada permitiu identificar, além das funcionalidades básicas, outras necessidades que estão presentes na maioria das aplicações multimídia da Internet e que podem ser atendidas através da existência de uma camada de sessão. Essas funcionalidades foram acrescentadas ao modelo por meio dos componentes Análise de Tráfego e Gerência de Buffer.

Finalmente, a extensão proposta oferece um serviço de controle de sessão multiponto (CSMu) que permite que um servidor de aplicação distribua o tráfego entre clientes dispersos na rede, utilizando os dados armazenados no buffer de player.

A aplicação dessa proposta pode ser demonstrada através dos casos de uso descritos no Apêndice A.

5 CONCLUSÕES

Buscando atender à demanda atual de computadores e equipamentos portáteis ligados à Internet, novos padrões foram desenvolvidos e acrescentados a esta, como, por exemplo: o IPv6 na camada de rede que permite QoS; a arquitetura WAP que adapta os equipamentos portáteis na rede mundial de computadores; o protocolo SIP que possibilita serviço de telefonia sobre rede IP, entre outros citados no contexto deste trabalho. Entretanto, desde a sua origem, a arquitetura da Internet nunca foi remodelada.

Este trabalho focalizou o problema de crescimento desenfreado da Internet, em que não houve acompanhamento simultâneo para rever sua arquitetura. O desenvolvimento desta tese de doutorado gerou uma proposta de extensão à arquitetura da Internet que permite amenizar o problema de redundância de código presente nas aplicações, agrupando as funcionalidades de sessão em uma camada específica e oferecendo serviços para facilitar o desenvolvimento de futuras aplicações.

5.1 Avaliação dos Resultados

O desenvolvimento desta tese atendeu aos objetivos propostos e culminou com alguns resultados importantes, que geraram alternativas de soluções para diversos problemas atuais.

No Apêndice B apresentam-se os serviços de sessão especificados no modelo de referência RM-OSI. Esta revisão bibliográfica foi utilizada no método comparativo que permitiu identificar e constatar a existência da camada de sessão na arquitetura da Internet, como descrito no Capítulo 2.

No Apêndice C apresenta-se uma descrição sobre a arquitetura da Internet e sobre o funcionamento do kernel do Unix. Embora a arquitetura da Internet seja independente da API de sockets do Unix, considerando que outros sistemas, como o Windows, não implementam a pilha TCP/IP dessa forma, esta abordagem foi escolhida por se tratar da primeira API desenvolvida e por haver uma farta documentação de como ela é implementada no Unix. A relevância na descrição desta API está na sua vasta utilização e nas considerações de como a camada de sessão poderia se encaixar nela. Um trabalho de conclusão de curso de computação foi orientado no decorrer dos anos de 2003 e 2004 a fim de programar um protótipo de um protocolo de sessão para a arquitetura TCP/IP denominado SESP (HENCHEN, 2004). Nesse

trabalho consideraram-se duas abordagens iniciais: a implementação no kernel do sistema operacional Unix (no espaço do sistema operacional) e a implementação em forma de bibliotecas (em espaço de usuário), a serem relacionadas com aplicações. Esse trabalho foi importante para verificar a viabilidade de se incluir uma camada de sessão na arquitetura da Internet.

O Apêndice A possui casos de uso (recurso da linguagem UML) que permitem identificar o comportamento e ações de um sistema que contempla a extensão proposta nesta tese. Utiliza-se como ator uma aplicação multimídia.

A identificação das redundâncias de serviços de sessão foi abordada no Capítulo 3, em que se apresenta a repetição de funcionalidades em grande parte das aplicações conhecidas na Internet. A extensão proposta pretende atender a qualquer tipo de aplicação, porém um direcionamento foi dado à área de multimídia quando se identificou que a demanda de recursos gerada por esse tipo de aplicação não estava prevista no projeto original da arquitetura TCP/IP. Por causa disso, acredita-se que investimentos direcionados a esse tipo de aplicação proporcionarão maior aceitação da extensão proposta na academia e demais organismos de pesquisa.

Assim como no Capítulo 3, uma dissertação de mestrado realizada por um integrante da mesma equipe (CUNHA, 2005) também teve como base uma pesquisa que identificava redundâncias de serviços das camadas de apresentação e sessão nas aplicações atuais.

Em artigo publicado em organismo internacional (Anexo A), cuja proposta é a inserção de uma camada de sessão entre as camadas de transporte e rede da arquitetura TCP/IP, os avaliadores consideraram a motivação válida, pois atualmente se tem uma idéia melhor dos diversos requisitos das aplicações multimídia do que se tinha há anos, no início da Internet. Também se abordou sobre a vantagem em liberar as aplicações de implementar requisitos comuns, que podem estar presentes numa camada de sessão.

Durante o período de levantamento bibliográfico em artigos científicos publicados em organismos internacionais (referenciados no item 3.6), identificaram-se problemas tais como a dificuldade de categorização de tráfego e metodologias de utilização de buffers, replicados em várias aplicações multimídia. Embora solucionar esses problemas não fosse o objetivo inicial desta tese, a extensão proposta apresenta duas formas para atender a essa necessidade, como abordado na seção 4.6, que descreve o funcionamento do componente Análise de Tráfego.

5.2 Contribuições do Trabalho

No escopo desta tese identificaram-se na seção 2.6 contribuições de caráter exploratório em que se utilizou o método comparativo, permitindo constatar que existe sessão na Internet, ao contrário do que é afirmado por alguns autores de renome. Além disto, têm-se as seguintes contribuições de caráter investigativo:

a) a seção 3.2 apresenta algumas soluções de organismos de pesquisa em busca de alternativas para categorizar o tráfego da Internet;
b) a seção 3.4 apresenta um quadro comparativo onde se constata a presença dos mesmos protocolos implementados dentro de várias aplicações, caracterizando o problema de redundância de código; e
c) a seção item 3.6 apresenta pesquisas destacando alterações nos protocolos de transporte, visando atender à demanda de aplicações multimídia.

Relacionado a esses fatos observa-se que existem propostas de soluções, algumas destas padronizadas, para os problemas de escassez de endereço IP e para os problemas relacionados aos retardos do protocolo de transporte TCP, tais como o IPv6 e o FastTCP. Dessa forma, na Internet 2, o meio físico, a camada de rede e a camada de transporte já contemplam alternativas para a demanda futura das aplicações multimídia. Esta tese contribui para a organização da pilha Internet quando propõe criar uma base de serviços de sessão para auxiliar o desenvolvimento de novas aplicações.

Para evitar que aplicações consolidadas precisassem ser adaptadas à extensão proposta, utilizou-se a facilidade de *pass-throught*, o que possibilita que a camada de sessão proposta ficasse transparente para as aplicações que já estão consolidadas no contexto da atual arquitetura da Internet.

Dois dos serviços propostos nesta tese apresentam contribuições de caráter inédito. O primeiro sugere alterar o conceito inicialmente utilizado na transmissão de dados multimídia, em que o tráfego é centralizado no servidor. Na extensão proposta, o tráfego é distribuído nas pontas pelos próprios clientes, e um mecanismo de gerência de conexões é centralizado no servidor. Essa forma, descrita na seção 4.5, permite um modelo de distribuição de tráfego cujo funcionamento não necessita da instalação de dispositivos de *proxy* em pontos específicos da rede. A camada de sessão é responsável pela distribuição desse tráfego, realizando a funcionalidade de sincronização dos buffers de player e oferecendo este serviço às aplicações. Tal idéia implementa o conceito de unidades de controle multiponto (MCU)

dentro da camada de sessão, em que a gerência de sessões é feita pelo servidor, que utiliza um modelo P2P (*peer to peer*) na distribuição do tráfego.

O segundo serviço de sessão refere-se à forma como a extensão proposta possibilita a divisão do tráfego em categorias. A seção 4.6 descreve duas alternativas para realizar a análise de tráfego: uma que possibilite monitorar o comportamento das aplicações existentes (abertura de sockets), tornando possível identificar o tipo de tráfego gerado por elas e, conseqüentemente, categorizá-lo; e outra, mais adequada, quando a aplicação utiliza a extensão proposta nesta tese, sinalizando o tipo de tráfego que ela irá gerar ao utilizar um serviço de sessão.

Finalmente, uma última contribuição refere-se às adaptações realizadas nos serviços de sessão propostos pelo RM-OSI. Alguns desses serviços precisaram ser adaptados ao contexto atual da Internet, como, por exemplo, o mecanismo de gerência de diálogo, que, na extensão proposta, torna possível implementar a idéia do moderador.

A proposta de extensão da arquitetura da Internet apresentada neste trabalho procurou sempre o caminho mais simples para o uso do conceito de sessão. Não se especializou a ponto de prever tudo o que as aplicações gostariam ou poderiam necessitar. Dessa forma, ela não é definitiva ou exaustiva; ao contrário, é flexível a ponto de permitir a adição de novos serviços, que possam ser identificados em trabalhos futuros, como necessários para o suporte a outras aplicações.

5.3 Trabalhos Futuros

Os trabalhos desenvolvidos nesta tese oferecem subsídios ao desenvolvimento de outros projetos de pesquisa, abaixo relacionados.

Com a utilização da extensão proposta é possível abstrair o conceito de portas utilizado pela camada de transporte e ater-se apenas ao conceito de sessões. Dessa forma, um trabalho futuro poderia propor um filtro IP, ferramenta de controle de acesso usualmente disponível em *firewalls*, que eleve o nível de filtragem para a camada de sessão. Ferramentas como IPTables, IPFilter (REED, 1999) ou IPFirewall poderiam ser alteradas, de forma a demonstrar a viabilidade da solução. Por exemplo, em vez de permitir a entrada do tráfego para a porta 8080 de um determinado servidor da Intranet, permite-se a passagem do tráfego de sessão cuja categoria for Web e a instância for Tomcat. Neste caso, quando se permite a entrada do tráfego multimídia, não interessa mais quantas portas de comunicação serão abertas, pois a

filtragem por sessões permite avaliar o tráfego por categoria, não sendo barrado mais pelos *firewalls* o uso de portas aleatórias.

Ainda com relação às implementações de mecanismos de *firewalls*, uma segunda alternativa pode ser considerada. Nesta, mantém-se a identidade de uma aplicação em função da porta padrão que ela utiliza, estendendo-se no filtro o mecanismo de acompanhamento de estado (*keep state*), que permite reconhecer o id_session gerado a partir do início de uma sessão ao se conectar com determinada porta. Por exemplo, um cliente pede uma sessão com uma aplicação Tomcat pela porta padrão 8080. Essa aplicação gera um id_session e transmite-o para o cliente. Um filtro no meio do caminho acompanha essa transação e armazena o id_session gerado. A partir dessa etapa, qualquer porta aleatória associada a esse id_session terá entrada permitida na Intranet. A Figura 27 demonstra essa alternativa.

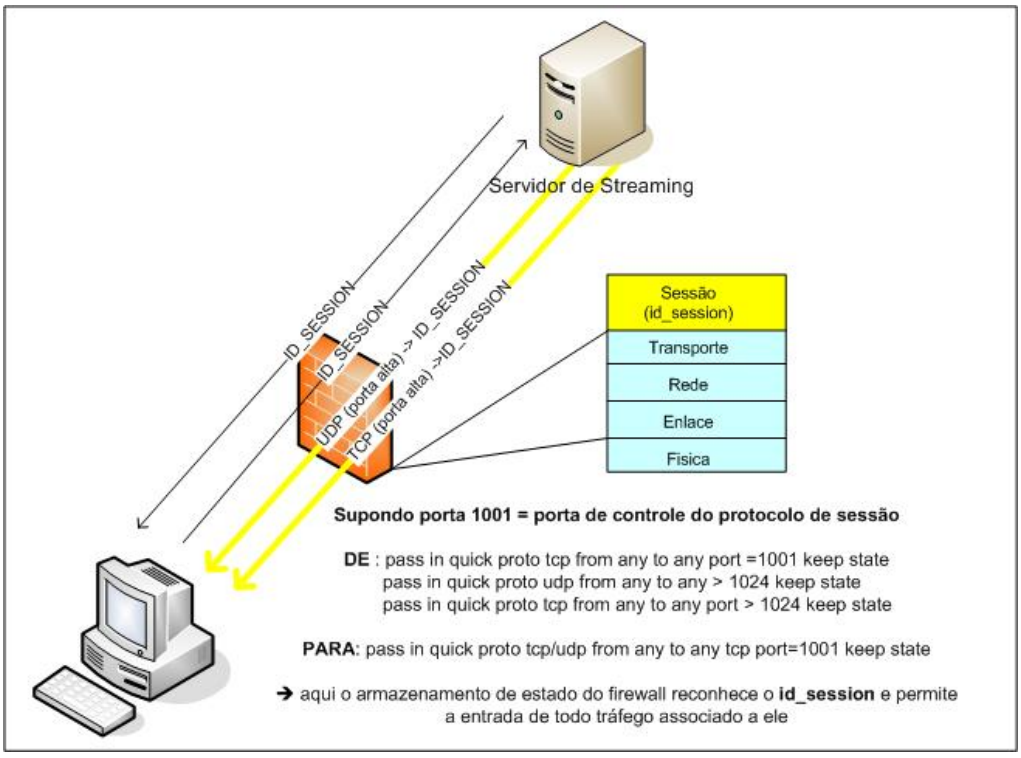


Figura 27 – Mecanismos de filtros baseados no id_session

As duas alterações propostas acima permitem a abstração do conceito de portas. Entre elas considera-se a última mais leve e de mais fácil aceitação, pois mantém a estrutura atual utilizada na rede (filtros baseados em endereços IP e portas padronizadas pelo IANA) e acrescenta a possibilidade de acompanhar o estado das sessões (*keep state*). Essa solução é mais próxima do funcionamento atual da arquitetura da Internet, uma vez que preserva os conceitos de padronização já utilizados por ela.

A extensão proposta provê uma forma não encriptada de determinar o identificador de sessão. Ao contrário do SSL, para simplificar a geração do identificador de sessão, não houve preocupação com a identidade do servidor. A exemplo do WAP, nesta extensão, a segurança deve ser modular e pode ser inserida abaixo da camada de transporte, através da utilização de protocolos de segurança, tais como o IPsec; ou acima, através da utilização do próprio SSL. Trabalhos futuros podem ser direcionados para a avaliação do impacto de segurança que se tem ao utilizar a extensão proposta com esses protocolos.

Não é função da camada de sessão garantir a integridade dos dados, ou seja, garantir que o dado recebido é exatamente o dado que foi enviado. Esta função está ligada à camada de apresentação. Assim como se encontrou nas aplicações da Internet a presença de sessão, o mesmo pode ser feito para a apresentação dos dados, e um segundo trabalho pode propor uma extensão à Internet contemplando uma camada de apresentação.

Na extensão proposta, todos (cliente e servidor) confiarão no id_session. Se alguém "seqüestrar" o id_session no meio do caminho, ele poderá ser utilizado de forma ilegítima. Essa possibilidade se agrava pelo fato de existir o problema de como vincular o id_session a um cliente, ou seja, testar a autenticidade do cliente. Tanto cliente quanto servidor devem testar o id_session conferindo o IP de origem das PDUs enviadas e recebidas. Essa forma de checagem bidirecional é feita pela camada de sessão, mas pode não ser suficiente para garantir a autenticidade tanto do cliente quanto do servidor.

Ainda na área de segurança, um outro trabalho pode avaliar os recursos necessários para inserir o Kerberos[17] dentro da extensão proposta, em que o identificador de sessão seriam os tíquetes. Acredita-se que esse método gera um *overhead* na camada de sessão, mas permite garantir a autenticidade do cliente e do servidor.

[17] "Kerberos é um padrão da Internet, desenvolvido originalmente pelo MIT, que é baseado em encriptação e utiliza abordagem de três pontas para autenticação: um banco de dados que contém os direitos dos usuários, um servidor de autenticação e um servidor de emissão de tíquetes" (GALLO; HANDCOOK, 2003).

6 REFERÊNCIAS BIBLIOGRÁFICAS

AGNEW, P. et al. **Distributed multimedia**. Reading, MA: Addison Wesley, 1996.

BARDIN, Laurence. **Análise de conteúdo**. Lisboa: Edições 70, 1977.

Berelson, B. **Content analysis.** In: Communication Research. New York: University Press, 1952.

BELLOVIN, S. **Firewall-Friendly FTP**, RFC 1579, 1994. Disponível em: <http://www.ietf.org/rfc/rfc1579.txt?number=1579>. Acesso em: 20 maio 2004.

BLAZE, Matt et al. **IP Security Policy (IPSP) Requirements**, RFC 3586, 2003. Disponível em: <http://rfc3586.x42.com/>. Acesso em: 4 jun. 2004.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML guia do usuário**. Editora Campus,1. ed, 2000. ISBN 85-352-0562-4

BORTOLUZZI, Dayna Maria. **Utilização de filtros de escalamento de mídia na interconexão entre duas redes heterogêneas**. 1999. Dissertação (Mestrado em Ciências da Computação) – Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis, 1999.

_____. **Uso de filtros adaptativos na conexão entre redes heterogêneas** 2000. VI Simpósio Brasileiro de Sistemas Multimídia e Hipermídia. Natal, RN.

_____. **An extended model for TCP/IP architecture.**2004. Disponível em: <http://www.i2r.a-star.edu.sg/iccs2004/>. Acesso em: 5 set. 2004.

BRADEN, Bob et al. **Resource ReSerVation Protocol (RSVP)**, RFC 2205, 1997. Disponível em: <http://rfc2205.x42.com/>. Acesso em: 20 maio 2004.

_____. **Recommendations on queue management and congestion avoidance in the internet**, RFC 2309, April (1998).

CADP, 2004. **Construction and Analysis of Distributed Processes**. Disponível em: <http://www.inrialpes.fr/vasy/cadp/#Introduction>. Acesso em: 5 set. 2004.

CALTECH , 2005. Califórnia Institute of Technology. Disponível em: <http://netlab.caltech.edu/pub/papers/fast-030401.pdf>. Acesso em: 5 março 2005.

CARDOSO, Olga Regina. **Foco da qualidade total de serviços no conceito do produto ampliado**. 1995. Tese (Doutorado em Engenharia de Produção) – Departamento de Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis, 1995.

CARPENTER, Brian E. **Resource Internet Transparency**, RFC 2775, 2000. Disponível em: <http://rfc2775.x42.com>. Acesso em: 20 maio 2004.

CHEN, Zhihua; BODENHEIMER, Bobby; BARNES, J. Fritz. **Robust Transmission of 3D Geometry over Lossy Networks.** Association for Computing Machinery, Inc, 2003.

CISCO SYSTEMS**. The Internet Protocol Journal - The Session Initiation Protocol, by William Stallings**. Volume 6, issue 1, march 2003. Disponível em: <http://www.cisco.com/warp/public/759/ipj_6-1/ipj_6-1_session_initiation_protocol.html>. Acesso em: 28 fev. 2005.

CROCKER, David H. **Standard for ARPA Internet Text Messages**, RFC 822, 1982. Disponível em: <http://www.ietf.org/rfc/rfc822.txt?number=822>. Acesso em: 4 jun. 2004.

CUNHA, Erivelto S. **Identificação de serviços de apresentação e sessão para a arquitetura da Internet.** Dissertação de Mestrado do Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis, 2005.

FIELDING, R.et al. **Hypertext Transfer Protocol – HTTP/1.1**. RFC 2616, 1999. Disponível em: <http://www.ietf.org/rfc/rfc2616.txt?number=2616> Acesso em: 28 ago. 2004.

FINLAYSON, Ross. **IP Multicast and Firewalls**. RFC 2588, 1999. Disponível em: <http://rfc2588.x42.com/>. Acesso em: 20 maio 2004.

FLOYD, S; FALL K. **Promoting the Use of End-to-End Congestion Control in the Internet**, IEEE/ACM Transations on Networking August (1999).

FREEBSD ARCHITECTURE Handbook, The. [S.l.]: **The FreeBSD Documentation Project**, 2003. Disponível em: <http://www.freebsd.org/doc/en_US.ISO8859-1/books/arch-handbook/index.html>. Aceso em: 23 nov. 2003.

FREEBSD DEVELOPER's Handbook, The. [S.l.]: **The FreeBSD Documentation Project**, 2003. Disponível em: <http://www.freebsd.org/doc/en_US.ISO8859-1/books/developers-handbook/index.html>. Aceso em: 23 nov. 2003.

GALLO, Michael A.; HANCOCK, William M. **Comunicação entre computadores e tecnologias de rede**. 1. ed. Thomson, 2003. ISBN 85-221-0293-7

GIL, Antonio Carlos. **Pesquisa social**. 5. ed. São Paulo: Atlas, 1999. ISBN 85-224-2270-2.

HAIAN, Tony. **Architectural Implications of NAT**, RFC 2993, 2000. Disponível em: <http://rfc2993.x42.com/>. Acesso em: 21 maio 2004.

HALSALL, Fred. **Data communications, computer networks and open systems**. 4. ed. Harlow, England: Addison-Wesley Publishing Company, 1997. ISBN 0-201-42293-X

HANDLEY, Mark; JACOBSON Van. **SDP: Session Description Protocol**, RFC 2327, 1998. Disponível em: <http://www.ietf.org/rfc/rfc2327.txt?number=2327>. Acesso em: 19 maio 2004.

_____. **SIP: Session Initiation Protocol**, RFC 2543, 1999. Disponível em: <http://rfc2543.x42.com/>. Acesso em: 20 maio 2004.

HENCHEN, Rafael. **Implementação de um protocolo de controle de sessão para aplicações TCP IP**. 2004. TCC (Graduação em Ciências da Computação) - Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis, 2004.

ITU-T. Telecomunication Standardization Sector. **Recommendation X.200: Open System Interconection – Basic Reference Model: The Basic Model**. [S.l.]: ITU-T, 1994.

ITU-T. Telecomunication Standardization Sector. **Recommendation X.215: Information Technolog y - Open System Interconection – Session Service Definition**. [S.l.]: ITU-T, 1995.

ITU-T. Telecomunication Standardization Sector. **Recommendation X.225: Information Technology - Open System Interconection – Connection-Oriented Session Protocol: Protocol Specification.** [S.l.]: ITU-T, 1995.

JACOBSON, V.; BRADEN, R. **TCP Extensions for Long-Delay Paths**, RFC 1072, 1988. Disponível em: <http://rfc1072.x42.com/>. Acesso em: 20 maio 2004.

JACOBSON, Van; BRADEN, R.; BORMAN, Dave. **TCP Extensions for High Performance**, RFC 1323, 1992. Disponível em: <http://rfc1323.x42.com/>. Acesso em: 20 maio 2004.

JENNINGS, Cullen; PETERSON, Jon; WATSON, Mark. **Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks**, RFC 3325, 2002. Disponível em: <http://rfc33253.x42.com/>. Acesso em: 24 maio 2004.

KAUSAR, Nadia; BRISCOE, Bob; CROWCROFT, Jon. **A charging model for sessions on the Internet.** SO Multimedia Applications, Services and techniques – ECMAST (1999) Department of Computer Science, University College London. Gower stret, London WCiE 6BT, UK.

KENT, Stephen; ATKINSON, Randall. **Security Architecture for the Internet Protocol**, RFC 2401, 1998. Disponível em: <http://www.ietf.org/rfc/rfc2401.txt?number=2401>. Acesso em: 15 fev. 2004.

KELLY, Scott; RAMAMOORTHI, Sankar. **Requirements for IPsec Remote Access Scenarios**, RFC 3457, 2003. Disponível em: <http://rfc3457.x42.com/>. Acesso em: 4 jun. 2004.

KRISTOL, D.; MONTULLI, L. **HTTP State Management Mechanism**, RFC 2109, 1997. Disponível em: <http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc2109.html.> Acesso em: 28 fev. 2005.

KURNIAWAN, Budi. **Java para web com servlets, JSP e EJB**: um guia do programador para soluções escalonáveis em J2EE. Rio de Janeiro: Ciência Moderna, 2002. ISBN 85-7393-210-4

LAKATOS, Eva Maria; MARCONI, Marina de Andrade. **Metodologia científica**. 2. ed. São Paulo: Atlas, 1992.

LU, Guojun. **Communication and computing for distributed multimedia systems**. 1 Ed. Boston, London: Artech House Inc., 1996

MAEDA, C.; BERSHAD, B. N. **Protocol service decomposition for high-performance networking**. In: Proceedings of the Fourteenth ACM Symposium on Operating Systems Principles, 1993, p. 244-255.

MERWE, Jacobus van der; CÁCERES, Ramón; CHU, Yang-hua; SREENAN, Cormac. Mmdump: **A Tool fo Monitoring Internet Multimedia Traffic**. Computer Communication Review ACM SIGCOMM (2000), volume 30, issue 5, p: 48-59. ISSN: 0146-4833. ACM Press New York, USA.

MMUSIC. **IETF Multiparty Multimedia Session Control Working Group**. Disponível em: <http:// www.ietf.org/html.charters/mmusic-charter.html>. Acesso em 20 set 2004.

SMETANA, George Marcel Monteiro Arcuri. **Um sistema de conferência centralizada com controle de posse da palavra para educação a distância**. Dissertação de Mestrado - Escola Politécnica da Universidade de São Paulo, São Paulo, 2004.

MULLER, Nathan J. Improving and Managing Multimedia Performance Over TCP/IP Nets. **International Journal of Network Management**, v. 8, p. 356-367, 1998.

OMA. **Open Mobile Alliance**. 2003. Disponível em: <http://www.wapforum.org/>. Acesso em 20 set 2004.

PETERSON, Larry L.; DAVIE, Bruce S. **Computer networks: a systems approach**. 2. ed. San Francisco: Morgan Kaufmann Publishers, 2000. ISBN 1-55860-514-2

PROENÇA. **Trabalho de TCP/IP e WAP**. Disponível em: <http://proenca.uel.br/curso-redes-graduacao/2000/trab-04/equipe-03/index.html>. Acesso em: 29 jun. 2003.

REED, Darren. **IP Filter**, Versão 3.5, 1993-1999. Disponível em: <http://coombs.anu.edu.au/~avalon/ip-filter.html>. Acesso em: 23 abr. 2004.

RESCORLA, Eric. **SSL and TLS designing and building secure systems**. Boston: Addison-Wesley Publishing Company, 2001. ISBN 0-201-61598-3

REYNOLDS, J.; POSTEL, J.; **Assigned Numbers** – RFC 1700. 1994. Disponível em: http://www.faqs.org/rfcs/rfc1700.html. Acesso em: 28 ago. 2004.

RNP, Rede Nacional de Ensino e Pesquisa. **CPqD e RNP inauguram rede óptica de altíssima velocidade.** 2004. Disponível em: <http://www.rnp.br/noticias/imprensa/2004/not-imp-040511.html>. Acesso em: 28 ago. 2004.

ROSENBERG, Jonathan et al. **SIP: Session Initiation Protocol**, RFC 3261, 2002. Disponível em: <http://www.ietf.org/rfc/rfc3261.txt?number=3261>. Acesso em: 19 maio 2004.

ROSENBERG, J.; Peterson, J; Schulzrinne, H. **Models for multi-party conferencing in SIP** IETF Internet Draft, nov. 2001, work in progress.

_____. **STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)**, RFC 3489, 2003. Disponível em: <http://rfc3489.x42.com/>. Acesso em: 4 jun. 2004.

ROSENBERG, Jonathan; SCHULZRINNE, Henning. **An Offer/Answer Model with the Session Description Protocol (SDP)**, RFC 3264, 2002. Disponível em: <http://www.ietf.org/rfc/rfc3264.txt?number=3264>. Acesso em: 18 maio 2004.

_____. **Reliability of Provisional Responses in the Session Initiation Protocol (SIP)**, RFC 3262, 2002. Disponível em: <http://www.ietf.org/rfc/rfc3262.txt?number=3262>. Acesso em: 21 maio 2004.

_____. **Session Initiation Protocol (SIP): Locating SIP Servers**, RFC 3263, 2002. Disponível em: <http://www.ietf.org/rfc/rfc3263.txt?number=3263>. Acesso em: 21 maio 2004.

SCHULZRINNE, Henning **RTP Profile for Audio and Video Conferences with Minimal Control**, RFC 1890, 1996. Disponível em: <http://asg.web.cmu.edu/rfc/rfc1890.html>. Acesso em: 18 maio 2004.

_____. **Requirements for Resource Priority Mechanisms for the Session Initiation Protocol (SIP)**, RFC 3487, 2003. Disponível em: <http://rfc3487.x42.com/>. Acesso em: 4 jun. 2004.

_____.Computer Science at Columbia University. **RTSP Implementations**, 2003. Disponível em: <http://www.cs.columbia.edu/~hgs/rtsp/implementations.html>. Acesso em: 18 maio 2004.

_____ et al. **RTP: A Transport Protocol for Real-Time Applications**, RFC 1889, 1996. Disponível em: <http://www.ietf.org/rfc/rfc1889.txt?number=1889>. Acesso em: 21 maio 2004.

_____ et al. **Real Time Streaming Protocol (RTSP)**, RFC 2326, 1998. Disponível em: <http://burks.brighton.ac.uk/burks/internet/rfcs/rfcs/26/rfc2326.htm>. Acesso em: 18 maio 2004.

SELLTIZ, Claire et al. **Métodos de pesquisa nas relações sociais**. São Paulo: Herder, 1967.

SEN, Subhabrata; GAO, Lixin; REXFORD, Jennifer; TOWSLEY, Don. **Optimal Patching Schemes for Efficient Multimedia Streaming.** UMASS CMPSCI Technical Report 99 – 22. University of Massachusetts.

SHARDA, Nalin. Multimedia Networks: Fundamentals and Future Directions. **Communications of the association for information systems**, v1, 1999.

SIMOM, Robert; SOOD, Arun; MUNDUR, Padmavathi. **Network service selection for distributed multimedia applications**. Department of Computer Science, George Mason University, 2000.

SIPCenter. **SIP Products and Services Directory**. Disponível em: <http://www.sipcenter.com/vsts/products.html/ >. Acesso em: 19 jan. 2005.

SHORE, Melinda. **Establishing Reachability Behind NATs**. <draft-shore-nat-reachability-00.txt>, work in progress, 2004.

SOARES, Luis Fernando G.; LEMOS, Guido; COLCHER, Sérgio. **Redes de computadores, das LANs MANs e WANs as redes ATM**. 2. ed. Rio de Janeiro: Campus, 1995. ISBN 85-7001-954-8.

SOUSA, Pedro; FREITAS, Vasco. **A framework for the development of tolerant real-time application** . Computer Networks and ISDN Systems 30 (1998) 1531- 1541.

SRISURESH, Pyda; EGEVANG, Kjeld Borch. **Traditional IP Network Address Translator (Traditional NAT)**, RFC 3022, 2001. Disponível em: <http://rfc3022.x42.com/>. Acesso em: 21 maio 2004.

SRIPANIDKULCHAI, Kunwadee; MAGGS Bruce; ZHANG Hui. **An Analysis of Live Streaming Workloads on the Internet**. IMC"04, October 25-27, 2004. ACM 1-58113-821-0/04/0010. 2004.

STALLINGS, William. **Cryptography and network security principles and practice**. 2. ed. New Jersey: Prentice Hall, 1999. ISBN 0-13-869017-0

_____. **High-speed networks TCP/IP and ATM design principles**. New Jersey: Prentice Hall, 1998. ISBN 0-13-525965-7

_____. **Integrated Services Architecture: The Next-generation Internet**. International Journal of Network Management, v.9, 38-43, 1999.

STEVENS, W. Richard. **Advanced programming in the UNIX enviroment**. Reading, Massachusetts: Addison-Wesley, 1. ed. [S.l.: s.n.], 1998. ISBN 0-201-56317-7

_____ .**Unix Network Programming: Networking APIs - Sockets and XTI**, Volume 1. 2. 3. ed. New Jersey: Prentice Hall PTR, 1998.

TANENBAUM, Andrew S. **Computer networks**. 3. ed. Prentice-Hall, 1996.

_____ **Redes de computadores**. 3. ed. Rio de Janeiro: Campus, 1997. ISBN 85-352-0157-2

_____, **Redes de computadores**. 4. ed. Rio de Janeiro: Campus, 2003. ISBN 85-352-1185-3

THOMAS, Stephen A. **SSL and TLS essentials securing the Web**. New York: Wiley Computer Publishing, 2000. ISBN 0-471-38354-6

WALDEN, Ralph. **The Transmission Control Protocol TCP**. Edwardsville: Southern Illinois University. Disponível em: <http://www.ee.siue.edu/~rwalden/networking/tcp.html>. Acesso em: 18 maio 2004.

WANG, S. Y. **Design and implementing a new type of transport-layer socket: the UDTCP socket case**, Computer communications, vol 27, num 3, 262-272 (11), 2004.

WAP-HTTP; **Wireless profiled HTTP**: Disponível em: <http://www.wapforum.org >, WAP-229-HTTP-20010329-a, versão 29-Mar-2001. Acesso em: 10 jan. 2005.

WAP-TCP; **Wireless Profiled TCP**: Disponível em: <http://www.wmlclub.com/docs/especwap2.0/ WAP-225-TCP-20010331-a.pdf >, WAP-225-TCP-20010331-a, versão 31-March-2001. Acesso em: 21 maio 2005.

WAP-TLS; **Wireless Application Protocol TLS Profile and Tunneling Specification**: Disponível em: <http://www.openmobilealliance.org/tech/affiliates/wap/wap-219-tls-20010411-a.pdf >, WAP-219-TLS-20010411-a, versão 11-April-2001. Acesso em: 21 maio. 2005.

WAP-WSP; **Wireless Application Protocol Wireless Session Protocol Specification** Disponível em: <http://www.wapforum.org>, WAP-2003-WSP, Approved Version 4-May-2000. Acesso em; 10 jan. 2005.

WRIGHT, Gary R.; STEVENS, W. Richard. **TCP/IP illustrated**, Volume 2: The Implementation. Massachusetts, EUA: Addison-Wesley Publishing Company, 1995. ISBN 0-201-63354-X.

VIDE. **Vídeo Development Initiative**. Disponível em: <http://www.vide.net/ >. Acesso em: 19 jan. 2005.

APÊNDICE A – ESPECIFICAÇÃO UML

A especificação apresentada nesse apêndice visa facilitar a compreensão da aplicação da camada de sessão proposta nesta tese.

Os diagramas apresentados neste apêndice são os diagramas de estado da linguagem *Unified Modeling Language* (UML), que permitem a visualização, documentação e especificação dos artefatos iniciais na extensão proposta à arquitetura da Internet. Além dos diagramas de estado, nos casos de uso são apresentados detalhes de funcionamento e exemplos de pseudocódigo.

Engenheiros e projetistas podem desenvolver diferentes tipos de algoritmos e implementações desde que sigam o modelo geral detalhado no Capítulo 4.

A.1 Estabelecimento de Conexão de Sessão.

Descrição: Este caso de uso descreve o pedido de estabelecimento de conexão de sessão.

Fonte e/ou documentos relacionados:

- Norma X225_1_NormaSessãoOSI_ITUT
- Norma X225_2_NormaSessãoOSI_ITUT
- Capítulo 4

Pendências: N/A

Ator ativo: Aplicação multimídia

Outros Atores: N/A

Pré-Condições: N/A

Fluxo-base:

1. A aplicação pede o estabelecimento de conexão de sessão através da primitiva s_connect.request (INT-01).
2. A camada de sessão recebe o pedido de estabelecimento de conexão de sessão executa (FCS-01) e envia request para a entidade par (segue o diagrama de estados (MA-01)).
3. A entidade par recebe o pedido da camada de transporte e executa os fluxos (FCS-02) e (FCS-03), monta PDU e envia a resposta para a entidade par.
4. Em FCS-04 o pedido de conexão foi aceito. Fim no caso de uso.
5. Em FCS-04 o pedido de conexão foi rejeitado. Fim no caso de uso.

Pós-Condição (Fluxo-base):

- A execução do passo 4 exclui a execução do passo 5 e vice-versa.
- Se a conexão foi aceita como resultado têm-se um identificador de sessão.

FCS-01 – Fluxo da Camada de Sessão

No passo 2 do fluxo-base, a camada de sessão realiza as seguintes as ações:

1. Armazena os dados enviados pela aplicação, através da primitiva s_connect, que pode conter o seguinte formato: (id_service, id_classes, id_categoria, id_instância, endereço do buffer de sincronização, endereço do buffer de player).
2. Abre um canal de transporte confiável, preenche a PDU de pedido de conexão com os seguintes campos: (IP local, porta TCP local, IP destino, porta TCP destino, id_service, id_classes, id_categoria, id_instancia).
3. Envia o request para a entidade par.
4. Aguarda resposta da entidade par (servidor).

Pós-Condição (FCS-01)

- A aplicação ou a camada de sessão pode cancelar a operação encerrando esse caso de uso.

FCS-02 – Fluxo da Camada de Sessão

Pré-Condição (FCS-02): A entidade par aceita pedidos de conexão de sessão através de uma porta TCP. Existe um socket aberto no estado *listen*.

No passo 3 do fluxo-base, a camada de sessão realiza as seguintes ações:

1. Recebe PDU de pedido de estabelecimento de sessão. Através da primitiva id_service identifica-se que é um pedido de estabelecimento de sessão.
2. Calcula identificador de sessão.
3. Valida id_session (não pode existir dois id_session iguais associados a um mesmo IP).
4. Verifica as classes de serviço (id_classes) que a camada pode atender.
5. Se o campo id_categoria estiver preenchido, preenche campo na tabela de sessões, indicando a categoria do tráfego.
6. Notifica a aplicação sobre os dados recebidos.
7. Aguarda resposta da aplicação (via SAP – Ponto de Acesso de Serviço de Sessão).

FCS-3 – Fluxo da Camada de Sessão

No passo 3 do fluxo-base, a camada de sessão realiza as seguintes ações:

1. Recebe resposta da camada de aplicação (via SAP)
2. Se conexão aceita então armazena dados na tabela de conexões (id_session, IP remoto, id_classes, id_categoria, id_instância, tempo_de_atividade, endereço do buffer de sincronização).
3. Monta PDU com a resposta e envia para a entidade par, contendo os seguintes dados:

(IP local, porta TCP local, IP destino, porta TCP destino, id_service, id_classes, id_session, tempo de atividade)

FCS-4 – Fluxo da Camada de Sessão

No passo 4 do fluxo-base, a camada de sessão realiza as seguintes ações:

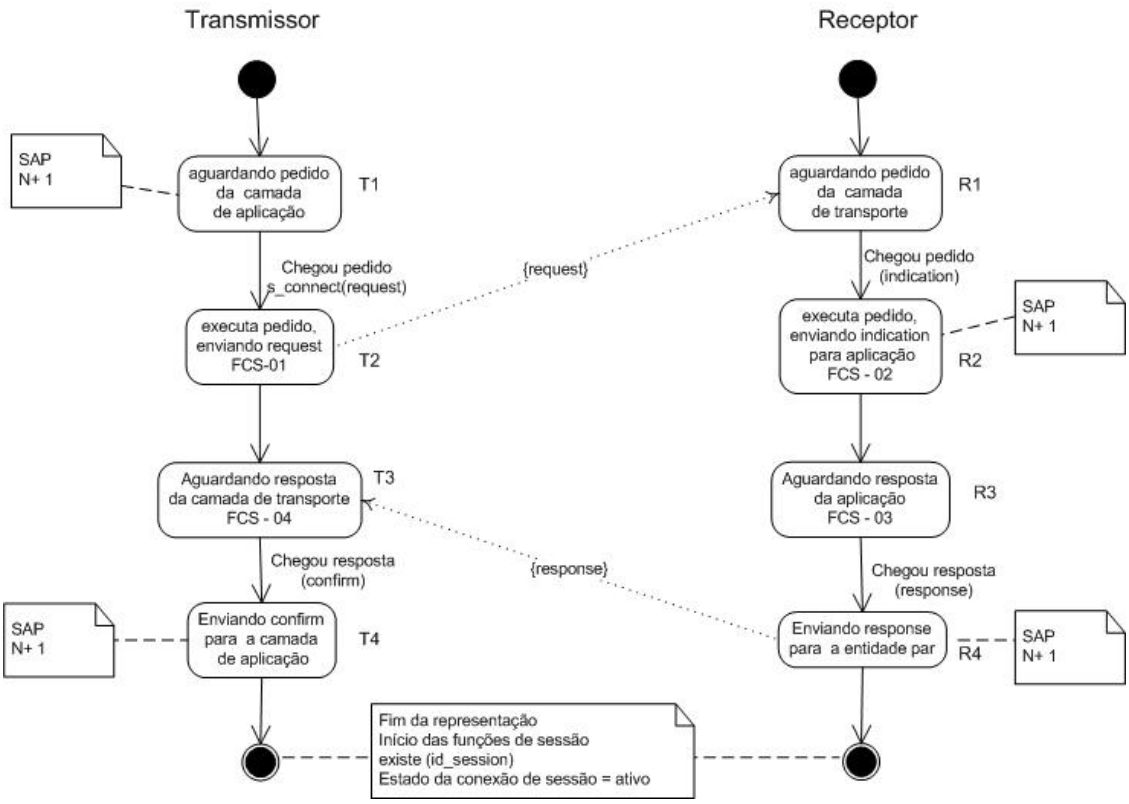
1. Recebe resposta da entidade par.
2. Desmonta PDU.
3. Verifica validade do id_session. Se campo nulo, notifica aplicação (s_aut_report) e vai para o passo 6.
4. Se a conexão foi aceita, então armazena dados na tabela de conexões (id_session, IP remoto, id_classes, id_categoria, id_instância, tempo_de_atividade, &buffer de sincronização, &buffer de player) e vai para o passo 6.
5. Se a conexão NÃO foi aceita por um dos motivos abaixo:
   a) recebeu um reject da entidade par -> sinaliza a aplicação através de um s_p_exception_report;
   b) o temporizador expirou e não veio resposta -> sinaliza a aplicação através de um s_u_exception _report;
   c) se o servidor não oferece um ou mais dos serviços requisitados -> sinaliza a aplicação através de um s_message_report.
6. Notifica a aplicação: aceita (4), ou rejeitada (5).
7. Termina caso de uso.

INT-01: Interface do Componente Controle de Sessão

PRIMITIVA	R	I	Rs	C	FUNÇÃO
Serviço Orientado a Conexão					
S_CONNECT S_CONNECT_M	• •	• •	• •	• •	**Estabelecimento de conexão** Estabelecimento de conexão *multicast*
S_RELEASE S_U_ABORT S_P_ABORT	• •	• • •	•	•	Liberação negociada de conexão Liberação abrupta (usuário) Liberação abrupta (fornecedor)

	S_U_EXCEPTION_REPORT **S_P_EXCEPTION_REPORT** **S_ MESSAGE_REPORT** **S_ AUT_REPORT**	• • •	• • • •			**Relatório de anomalia (usuário)** **Relatório de anomalia (fornecedor)** **Mensagem imprópria** **Falha na autenticação do id_session**	

MA-01 : Maquina de Estados



Estados	**Descrição**	**Evento de ativação (condição)**	**Ação**
Ocioso	Estado inicial da camada de sessão, quando não existe conexão de sessão	Ao ligar o sistema Ao receber a primitiva s_connect (reject). response Ao receber um pedido de liberação de conexão de sessão confirmado (MA-002)	Aguardando pedido de estabelecimento de conexões de sessão
Executa Pedido	Executando pedidos de conexão de sessão	s_connect.request s_connect.indication	Lado transmissor: FCS-01 Lado receptor: FCS-02
Aguarda Resposta	Aguarda resposta do pedido de estabelecimento de sessão	Recebe pedido de serviço confirmado	Lado receptor: FCS-03 Lado transmissor: FCS-04
Ativo	Existe uma ou mais conexões de sessão ativas	s_connect(accept).confirm	Estabelece conexão, armazena dados da conexão, executa demais funcionalidades da camada de sessão

A.2 Liberação Negociada da Sessão

Descrição: Este caso de uso descreve o pedido de liberação negociada de sessão.

Fonte e/ou documentos relacionados:
- Norma X225_1_NormaSessãoOSI_ITUT
- Norma X225_2_NormaSessãoOSI_ITUT
- Capítulo 4 e Apêndice B.

Pendências: N/A

Ator ativo: Aplicação multimídia

Outros Atores: N/A

Pré-Condições:

- Existe uma sessão estabelecida e um id_session associado a ela.

Fluxo-base 1:

1. A aplicação pede a liberação: s_release (request), (INT-01).
2. A camada de sessão recebe o pedido de liberação de conexão de sessão e executa FCS-05, e notifica e entidade par, seguindo o fluxo do diagrama de estados (MA-02).
3. A entidade par recebe pedido através da camada de transporte, executa os fluxos FCS-06 e FCS-07.
4. A sessão foi liberada de uma forma confirmada (FCS-08).
5. Fim no caso de uso.

Pós-Condição (Fluxo-base):

- Se o pedido de liberação de conexão de sessão foi aceito, a sessão não poderá ser resgatada.

FCS-5 – Fluxo da Camada de Sessão

No passo 2 do fluxo-base na camada de sessão, as seguintes ações são realizadas:

1. Recebe s_release.request (id_session);
2. Identifica os dados da sessão (IP remoto) na tabela de sessões;
3. Monta PDU e envia para a entidade par (id_session, id_service);
4. Aguarda resposta, pois o serviço de sessão é confirmado.

FCS-6 – Fluxo da Camada de Sessão

No passo 3 do fluxo-base, a camada de sessão realiza as seguintes ações:

1. Desmonta PDU (id_session, id_service);
2. Identifica dados da sessão na tabela de sessões;
3. Notifica a aplicação;
4. Aguarda resposta da aplicação, pois o serviço é confirmado.

FCS-7 – Fluxo da Camada de Sessão

No passo 3 do fluxo-base, a camada de sessão realiza as seguintes ações:

1. Chegou resposta da aplicação.
2. Se conexão liberada, então apaga registro desse id_session da tabela de sessões.
3. Se conexão não liberada, então não apaga nada;.
4. Monta PDU com a resposta da aplicação e envia a entidade par.

Pós-Condição (FCS-7):

- O passo 2 exclui a execução do passo 3 e vice-versa.

FCS-8 – Fluxo da Camada de Sessão

No passo 4 do fluxo-base, a camada de sessão realiza as seguintes ações:

1. Chegou a resposta da entidade par.
2. Se conexão liberada, então apaga registro deste id_session da tabela de sessões.
3. Se conexão não liberada, então não apaga nada.
4. Informa a aplicação da resposta recebida.

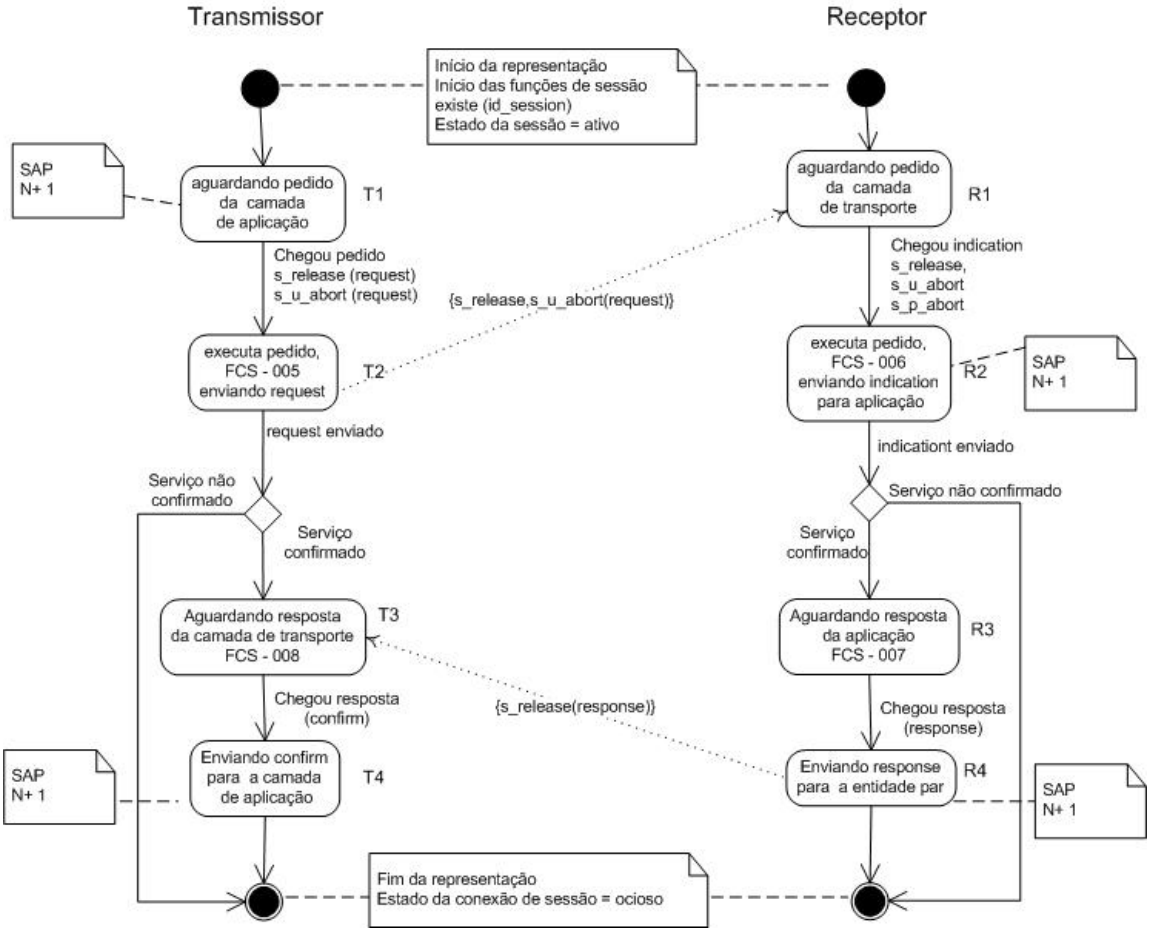
Pós-Condição (FCS-8)

- O passo 2 exclui a execução do passo 3 e vice-versa.

INT-01: Interface do Componente Controle de Sessão.

PRIMITIVA	R	I	Rs	C	FUNÇÃO
Serviço Orientado a Conexão (sobre TCP)					
S_CONNECT	•	•	•	•	Estabelecimento de conexão
S_CONNECT_M	•	•	•	•	Estabelecimento de conexão *multicast*
S_RELEASE	•	•	•	•	**Liberação negociada de conexão**
S_U_ABORT	•	•			Liberação abrupta (usuário)
S_P_ABORT		•			Liberação abrupta (fornecedor)
S_U_EXCEPTION_REPORT	•	•			Relatório de anomalia (usuário)
S_P_EXCEPTION_REPORT		•			Relatório de anomalia (fornecedor)
S_ MESSAGE_REPORT	•	•			Mensagem imprópria
S_ AUT_REPORT	•	•			Falha na autenticação do id_session

MA-02 : Maquina de Estados



Estados	Descrição	Evento de ativação	Ação
Ativo	A sessão existe	Ao receber uma das primitivas: s_connect (accept). response s_connect (accept). confirm	Executa funcionalidades de sessão Aguarda pedido finalização de conexão de sessão.
Executa Pedido	Executando pedidos de liberação de conexão de sessão	s_release_request, s_release_indication	FCS-05. FCS-06.
Aguarda Resposta	Aguarda resposta sobre o pedido de liberação de conexão de sessão	Recebe pedido de serviço negociado (com confirmação)	Aguarda resposta da aplicação FCS-07; Aguarda resposta da entidade par FCS-08.
Ocioso	Não existe mais sessão	s_release (accept) response s_release (accept). confirm	Libera recursos da sessão, apaga dados da sessão armazenados.

A.3 Liberação Abrupta de Sessão

Descrição: Este caso de uso descreve o pedido não confirmado de liberação de sessão, que pode ocorrer devido a uma falha de comunicação entre entidades pares, ou finalização inesperada da aplicação no lado cliente ou servidor.

Fonte e/ou documentos relacionados:

- Norma X225_1_NormaSessãoOSI_ITUT;
- Norma X225_2_NormaSessãoOSI_ITUT;
- Capítulo 4 e Apêndice B.

Pendências: N/A

Ator ativo: Aplicação multimídia.

Outros Atores: N/A

Pré-Condições:

- Existe uma sessão estabelecida, e um id_session associado a ela.

Fluxo-base 1:

1. A entidade par cliente aborta sessão: s_u_abort.request (INT-01). A camada de sessão executa FCS-05 e FCS-06.
2. A entidade par servidor aborta sessão: s_p_abort.indication (INT-01). A camada de sessão executa FCS-06.
3. A sessão foi abortada, inicia contabilização de tempo.

Pós-Condição (Fluxo-base):

- A sessão pode ser resgatada pela camada, pois seu registro não é imediatamente apagado.
- A sessão será finalizada caso temporizador expire (timer > tempo de atividade).

FCS-5 – Fluxo da Camada de Sessão

No passo 1 do fluxo-base, na camada de sessão, as seguintes ações são realizadas:

1 Recebe s_u_abort.request (id_session).
2 Identifica os dados da sessão (IP remoto, tempo de atividade) na tabela de sessões.
3 Monta PDU e envia para a entidade par (id_session, id_service).
4 Inicia contabilização de tempo (timer).
5 Marca (através de um FLAG) a sessão como inativa;
6 Serviço não confirmado. Segue diagrama de estados MA 02.

FCS-6 – Fluxo da Camada de Sessão

No passo 2 do fluxo-base, a camada de sessão realiza as seguintes ações:

1 Se recebeu s_u_abort.request
  a) Desmonta PDU (id_session, id_service).
  b) Identifica dados da sessão na tabela de sessões (tempo de atividade).
  c) Sinaliza aplicação (indication).
  d) Inicia contabilização de tempo (timer).

e) Marca (através de um FLAG) a sessão como inativa.

2 Se recebeu s_p_abort.indication
a) Identifica dados da sessão na tabela de sessões (tempo de atividade).
b) Sinaliza aplicação (indication).
c) Inicia contabilização de tempo (timer).
d) Marca (através de um FLAG) a sessão como inativa.

3 Serviço não confirmado. Segue diagrama de estados MA-02.

PCB-1 – Pós-condição do Fluxo-base

FCS-10 – Recuperação de uma sessão inativa

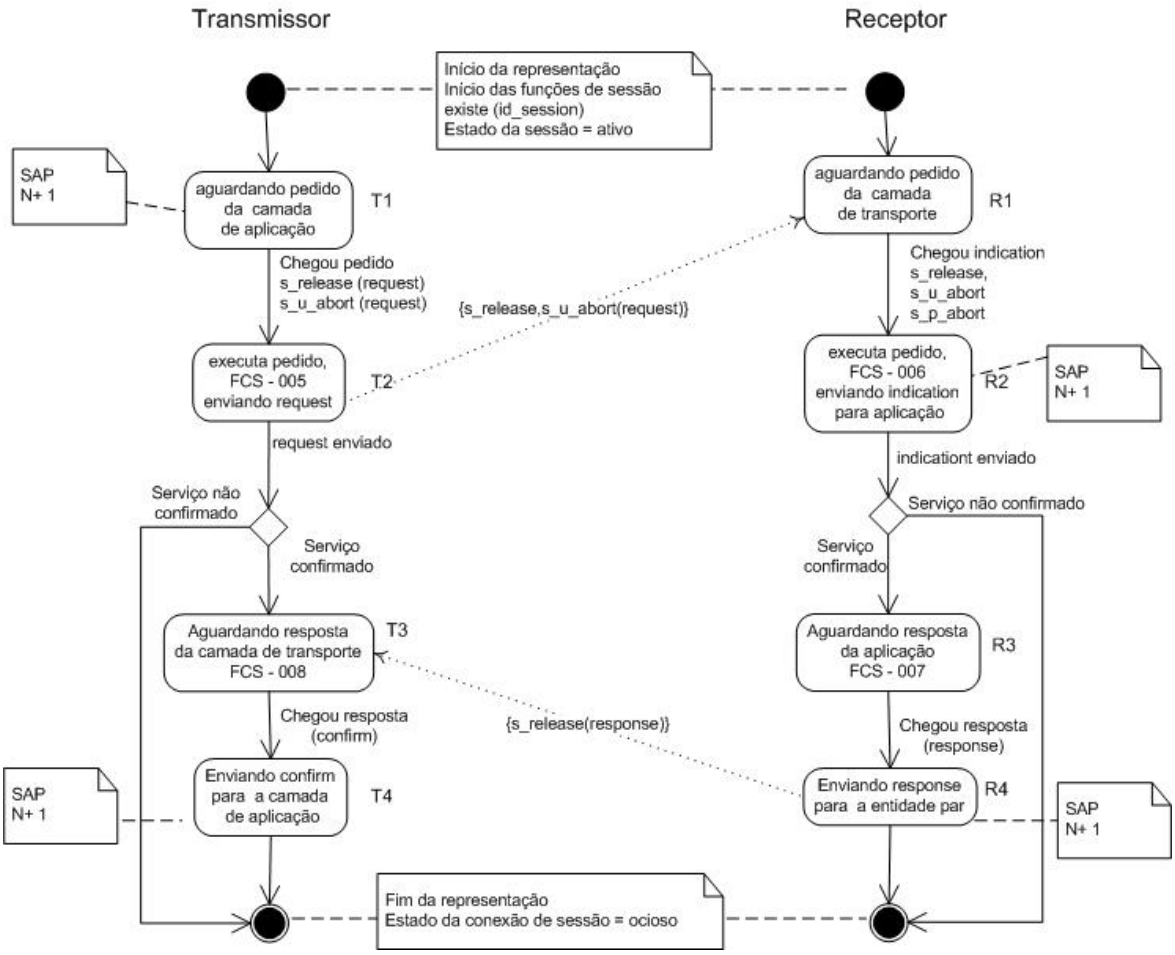
1 A camada de sessão recebe um uma PDU cujo id_session está marcado como inativo.
2 A camada de sessão obtém dados da sessão na tabela de sessões (tempo de atividade).
3 Se (timer < tempo de atividade) executa recuperação da sessão:
Zera temporizador (timer).
Marca (através de um FLAG) a sessão como ativa.

6. Se (timer > tempo de atividade) executa finalização da sessão:
Libera recursos daquela sessão (apaga registro id_session da tabela de sessões).
Sinaliza aplicação através de um s_aut_report.

INT-01: Interface do Componente Controle de Sessão.

PRIMITIVA	**R**	**I**	**Rs**	**C**	**FUNÇÃO**
Serviço Orientado a Conexão (sobre TCP)					
S_CONNECT	•	•	•	•	Estabelecimento de conexão
S_CONNECT_M	•	•	•	•	Estabelecimento de conexão *multicast*
S_RELEASE	•	•	•	•	Liberação negociada de conexão
S_U_ABORT	•	•			**Liberação abrupta (usuário)**
S_P_ABORT		•			**Liberação abrupta (fornecedor)**
S_U_EXCEPTION_REPORT	•	•			Relatório de anomalia (usuário)
S_P_EXCEPTION_REPORT		•			Relatório de anomalia (fornecedor)
S_ MESSAGE_REPORT	•	•			Mensagem imprópria
S_ AUT_REPORT	•	•			**Falha na autenticação do id_session**

MA-02 : Maquina de Estados



Estados	Descrição	Evento de ativação	Ação
Ativo	A sessão existe	Ao receber uma das primitivas: s_connect (accept). response s_connect (accept). confirm	Executa funcionalidades de sessão Aguarda pedido finalização de conexão de sessão.
Executa Pedido	Executando pedidos de liberação de conexão de sessão	s_release_request, s_release_indication	FCS-05. FCS-06.
Aguarda Resposta	Aguarda resposta sobre o pedido de liberação de conexão de sessão	Recebe pedido de serviço negociado (com confirmação)	Aguarda resposta da aplicação FCS-07. Aguarda resposta da entidade par FCS-08.
Ocioso	Não existe mais sessão	s_release (accept) response s_release (accept). confirm	Libera recursos da sessão, apaga dados armazenados da sessão.

A.4 Envio de Dados

Descrição: Este caso de uso descreve o envio de dados da camada de sessão, que pode utilizar um serviço de sessão confirmado (sobre TCP) ou não confirmado (sobre UDP).

Fonte e/ou documentos relacionados:

- Norma X225_1_NormaSessãoOSI_ITUT
- Norma X225_2_NormaSessãoOSI_ITUT
- Capítulo 4 e Apêndice B.

Pendências: N/A

Ator ativo: Aplicação multimídia.

Outros Atores: N/A

Pré-Condições:

- Existe uma conexão de sessão estabelecida (id_session).

Fluxo-base:

1. A aplicação tem dados para transmitir e o faz via primitivas de serviço (INT-02).
2. A camada de sessão (transmissor) executa FCS-10, lê os dados do buffer de transmissão e segue o fluxo de MA-03.
3. A camada de sessão (receptor) recebe os dados, executa FCS-11, alimenta buffer de recepção e sinaliza aplicação.
4. Se o serviço for confirmado (receptor) aguarda resposta da aplicação FCS-12 e envia a entidade par.
5. Se o serviço for confirmado (transmissor) recebe resposta da entidade par e executa FCS-13.
6. Fim do caso de uso.

Pós-Condição (Fluxo-base):

- A escolha da primitiva de serviço define o serviço de sessão (sobre UDP ou TCP) que a aplicação deseja utilizar.
- Os passos 4 e 5 do fluxo-base só são utilizados para dados de capacidade.

FCS-10 – Fluxo da Camada de Sessão

No passo 2 do fluxo-base a camada de sessão realiza as seguintes ações:

1. A camada de sessão recebe primitiva de serviço, identifica a aplicação (PID do processo no Unix), verifica se para esse id_session a classe de serviço solicitada (id_classes) foi negociada durante o estabelecimento da conexão de sessão.
   Se não foi: o componente CS notifica erro -> s_message_report
   Fim do caso de uso.
2. Através da primitiva de serviço utilizada a camada de sessão:
   a) Se serviço confirmado -> abre socket TCP (pode também usar um socket aberto); ou
   b) Se serviço não confirmado -> abre socket UDP (pode também usar um socket aberto).
3. Recebe dados.
4. Monta PDU (id_session, tamanho, id_service, dados).
5. Envia os dados através do socket aberto.
6. Se id_service = s_capability_data então aguarda resposta da aplicação.

Pós-Condição (FCS-10)

- Os passos 3,4 e 5 são repetidos enquanto existir dados para enviar.

FCS-11 – Fluxo da Camada de Sessão

No passo 3 do fluxo-base, a camada de sessão realiza as seguintes ações:

1 Através da camada de transporte (socket), a camada de sessão recebe uma PDU.
2 Desmonta PDU (id_session,tamanho,id_service,dados).
3 A camada de sessão pega o id_session e verifica na tabela de sessões, se essa classe (id_classe) foi negociada no estabelecimento da conexão de sessão.
Se não foi: o componente CS notifica erro -> s_message_report
Fim do caso de uso.
4 Entrega dados para aplicação (indication).
5 Se id_service = s_capability_data então aguarda resposta da aplicação.

Pós-Condição (FCS-11)

- Os passos 1, 2 e 4 são repetidos enquanto estiver chegando dados.

FCS-12 – Fluxo da Camada de Sessão

Pré-Condições: Esse fluxo só é utilizado na transferência de dados de capacidade (s_capability_data).

No passo 4 do fluxo-base, a camada de sessão realiza as seguintes ações:

1 Aguarda resposta da aplicação.
2 A aplicação confirma o recebimento dos dados.
3 Monta PDU de resposta: (id_session, tamanho, id_service, data).
4 Envia resposta para a entidade par.

FCS-13 – Fluxo da Camada de Sessão

Pré-Condições: Esse fluxo só é utilizado na transferência de dados de capacidade (s_capability_data).

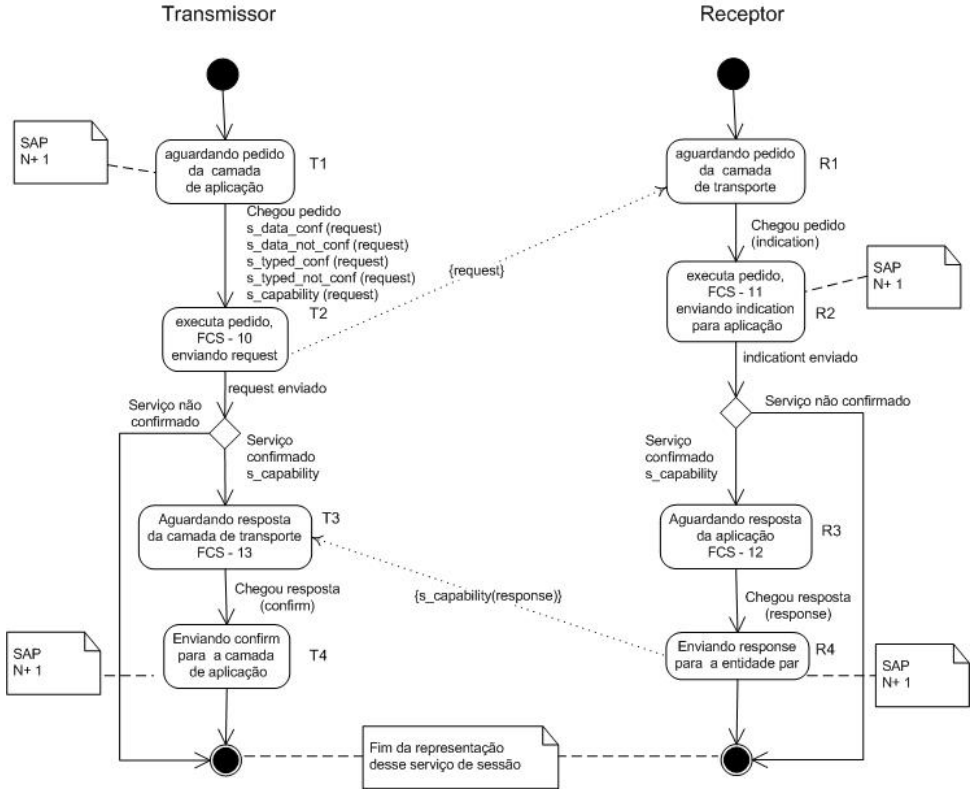
No passo 5 do fluxo-base, a camada de sessão realiza as seguintes ações:

1 Recebe resposta da camada de transporte.
2 Desmonta PDU: (id_session,tamanho,id_service,data).
3 Envia resposta para a aplicação.

INT-02: Interface da Camada de Sessão

PRIMITIVA	R	I	Rs	C	FUNÇÃO
Serviço com Sessão					
S_DATA_CONF	•	•			Serviço de transferência de dados normais confirmado (sobre TCP).
S_DATA_NOT_CONF	•	•			Serviço de transferência de dados normais não confirmado (sobre UDP).
S_TYPED_DATA_CONF	•	•			Serviço de transferência de dados classificados confirmado (sobre TCP)
S_TYPED_DATA_NOT_CONF	•	•			Serviço de transferência de dados classificados não confirmado (sobre UDP).
S_CAPABILITY_DATA	•	•	•	•	Serviço de transferência de dados de capacidade (apenas sobre TCP).

MA-03: Maquina de Estados



Estados	Descrição	Evento de ativação	Ação
Ativo	A conexão de sessão existe	Ao receber uma das primitivas: s_connect (accept). response s_connect (accept). confirm	Executa funcionalidades de sessão. Aguarda pedido finalização de conexão de sessão.
Executa Pedido	Executando pedido de transferência de dados	s_data_conf s_data_not_conf s_typed_data_conf s_typed_data_not_conf s_capability_data	Lado transmissor (FCS-10). Lado receptor (FCS-11).
Aguarda resposta	Apenas para dados de capacidade	Serviço confirmado também em nível de Camada de Sessão	Lado receptor (FCS-12). Lado transmissor (FCS-13).

A.5 Função de Sincronização

Descrição: Este caso de uso descreve a funcionalidade de sincronização da camada de sessão.

Fonte e/ou documentos relacionados:

- Norma X225_1_NormaSessãoOSI_ITUT
- Norma X225_2_NormaSessãoOSI_ITUT
- Capítulos 3, 4 e Apêndice B.

Pendências: N/A

Ator ativo: Aplicação multimídia.

Outros Atores: N/A

Pré-Condições:

- Existe uma conexão de sessão estabelecida (id_session).
- O serviço de sincronização foi negociado durante a fase de estabelecimento de sessão (id_classes).
- Existe um buffer de sincronização, cuja área foi criada pela aplicação, e o endereço desse buffer foi passado á camada de sessão (&buffer_sync).
- O buffer_sync contém uma cópia dos dados mais recentes, enviados (buffer de transmissão) ou recebidos (buffer de recepção).

Fluxo-base 1: Inserção de um Ponto de Sincronização.

1. A aplicação solicita a adição de um ponto de sincronização (INT-04).
2. A camada de sessão (transmisssor) executa FCS-14 e segue fluxo (MA-04).
3. A camada de sessão (receptor) recebe solicitação e FCS-15.
4. A camada de sessão (receptor) aguarda resposta da aplicação FCS-16, e envia a entidade par.
5. A camada de sessão (transmissor) recebe resposta e executa FCS-17.
6. Fim do caso de uso

Pós-Condição (Fluxo-base 1):

- Existe um ponto de sincronismo.
- A camada de sessão gerencia a ocupação do buffer_sync.

FCS-14 – Fluxo da Camada de Sessão

No passo 2 do fluxo-base 1, a camada de sessão realiza as seguintes ações:

1. Recebe dados da aplicação (pedido de sincronização).
2. Monta PDU (id_session, tamanho, id_service).
3. Envia os dados e aguarda resposta.

–

FCS-15 – Fluxo da Camada de Sessão

No passo 3 do fluxo-base 1, a camada de sessão realiza as seguintes ações:

1. Recebe dados da camada de transporte.
2. Desmonta PDU (id_session, tamanho, id_service).
3. Envia os dados para aplicação e aguarda resposta.

FCS-16 – Fluxo da Camada de Sessão

No passo 4 do fluxo-base 1, a camada de sessão realiza as seguintes ações:

1. Recebe resposta da aplicação.
2. Inserção de um novo ponto de sincronização.
3. Armazena posição atual do buffer_sync (B):
   sendo B [i] = posição dos dados no buffer; pto = i;
4. Os dados anteriores a esse ponto podem ser descartados.
5. Monta PDU de resposta (id_session, tamanho, id_service).
6. Envia PDU para a entidade par.

FCS-17 – Fluxo da Camada de Sessão

No passo 4 do fluxo-base 1, a camada de sessão realiza as seguintes ações:

1 Recebe resposta da aplicação.
2 Inserção de um novo ponto de sincronização.
Armazena posição atual do buffer_sync (B):
Sendo B [i] = posição dos dados no buffer; pto = i;
3 Os dados anteriores a esse ponto podem ser descartados.
4 Sinaliza aplicação (confirm).

Fluxo-base 2: Pedido de Re-sincronização.

1 A aplicação solicita o pedido de re-sincronização (INT-04).
2 A camada de sessão (transmisssor) executa FCS-14 e segue fluxo do MA-04.
3 A camada de sessão (receptor) recebe solicitação e FCS-15.
4 A camada de sessão (receptor) aguarda resposta da aplicação FCS-16, e envia a entidade par.
5 A camada de sessão (transmissor) recebe resposta e executa FCS-17.
6 Fim do caso de uso.

Pós-Condição (Fluxo-base 2):

- Os dados foram sincronizados.
- A camada de sessão gerencia a ocupação do buffer_sync.

FCS-14 – Fluxo da Camada de Sessão

No passo 2 do fluxo-base 2, a camada de sessão realiza as seguintes ações:

1 Recebe dados da aplicação (pedido de re-sincronização).
2 Monta PDU (id_session, tamanho, id_service).
3 Envia os dados e aguarda resposta.

FCS-15 – Fluxo da Camada de Sessão

No passo 3 do fluxo-base 2, a camada de sessão realiza as seguintes ações:

1 Recebe dados da camada de transporte.
2 Desmonta PDU (id_session, tamanho, id_service).
3 Envia os dados para aplicação e aguarda resposta.

FCS-16 – Fluxo da Camada de Sessão

No passo 4 do fluxo-base 2, a camada de sessão realiza as seguintes ações:

1 Recebe resposta da aplicação.
2 Desloca Buffer. Buffer de transmissão recebe dados do buffer de sincronização (B):
Sendo “i” ponto de transmissão do dado atual, e “pto” último ponto de sincronismo.
Buffer de transmissão é alimentado com dados de B [pto] até B[i].
3 Monta PDU de resposta (id_session, tamanho, id_service).
4 Envia a entidade par..

FCS-17 – Fluxo da Camada de Sessão

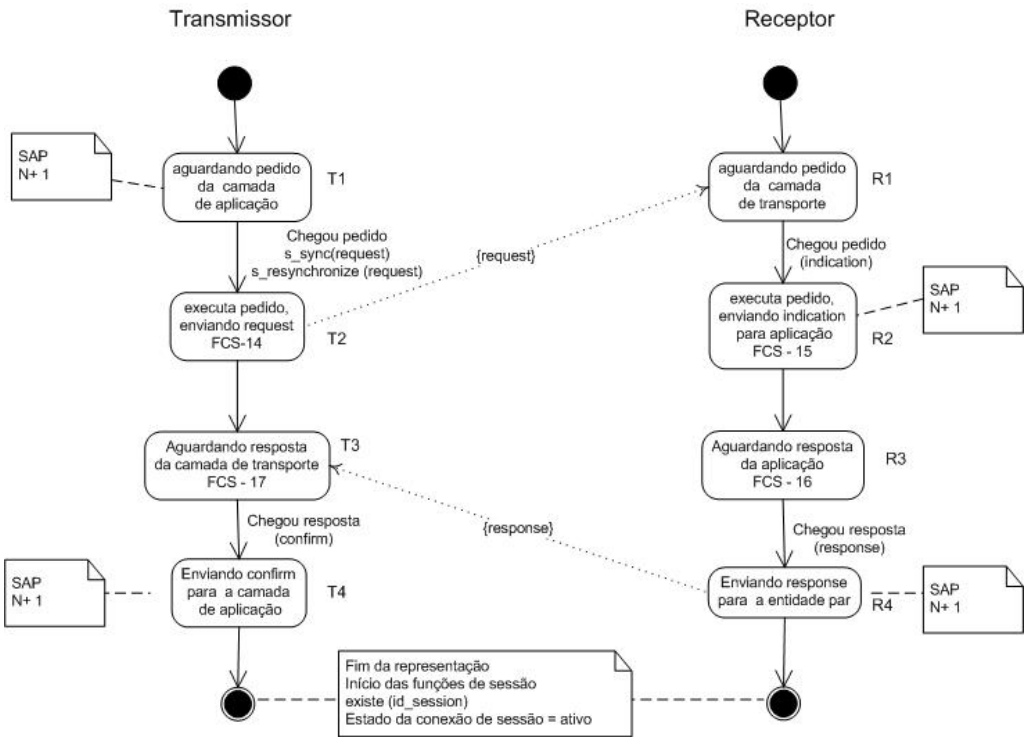
No passo 4 do fluxo-base, a camada de sessão realiza as seguintes ações:

1 Recebe resposta da aplicação;
2 Sinaliza aplicação (confirm).

INT-04: Interface da Camada de Sessão

PRIMITIVA	R	I	Rs	C	FUNÇÃO
Serviço com Sessão Confirmado					
S_SYNC	•	•	•	•	Inserção de ponto de sincronização
S_RESYNCHRONIZE	•	•	•	•	Pedido de resincronização

MA-04 : Diagrama de Estados do Serviço de Sincronização



Estados	Descrição	Evento de ativação	Ação
Ativo	A conexão de sessão existe	Ao receber uma das primitivas: s_connect (accept). response s_connect (accept). confirm	Executa funcionalidades de sessão Aguarda pedido finalização de conexão de sessão.
Executa Pedido	Executando pedido de inserção de ponto de sincronismo e de re-sincronização	s_sync_request, s_resynchronize_request, s_sync_indication, s_resynchronize_indication	
Aguarda Resposta	Aguarda resposta da entidade par, sobre o pedido de inserção de ponto de sincronismo ou re-sincronização	Recebe pedido de serviço negociado (com confirmação)	Monta PDU, envia para a camada de transporte e aguarda resposta da entidade par; Recebe PDU da camada de transporte, sinaliza a aplicação e aguarda resposta da aplicação.
Ativo	A função de sincronização foi realizada	Chegou a resposta de confirmação da entidade par s_sync_response s_sync_confirm s_resynchronize_response s_resynchronize_confirm	Insere novo ponto de sincronismo no buffer de dados, descarta dados antigos, sinaliza aplicação.

A.6 Serviço de Controle de Diálogo

Descrição: Este caso de uso descreve o serviço de controle de diálogo da camada de sessão

Fonte e/ou documentos relacionados:

- Norma X225_1_NormaSessãoOSI_ITUT
- Norma X225_2_NormaSessãoOSI_ITUT
- Capítulo 4 e Apêndice B.

Pendências: N/A

Ator ativo: Aplicação multimídia.

Outros Atores: N/A

Pré-Condições:

- Existe uma conexão de sessão *multicast* estabelecida (id_sessionCSMu).
- O mecanismo de gerência de diálogo utiliza um serviço de sessão confirmado (socket TCP).
- A entidade de gerou o id_sessionCSMu (servidor) é responsável por gerenciar o diálogo, descrito nesse caso de uso como moderador (receptor/transmissor).
- O serviço de gerência de diálogo foi negociado durante a fase de estabelecimento de sessão (id_classes).

Fluxo-base:

1. A aplicação, através da interface entre camadas, deseja obter a posse do token (INT-05).
2. A camada de sessão recebe a requisição e envia para a entidade par (moderador) o pedido de posse de token (FCS-18).
3. O moderador recebe o pedido (FCS-19) e sinaliza aplicação.
4. O moderador recebe resposta da aplicação e executa a função de gerência de diálogo (FCS-20).
5. Os participantes são notificados (FCS-21) e (FCS-22).

Pós-Condição (Fluxo-base):

- O moderador sempre sabe qual entidade detém o token. Caso essa entidade não responda (time-out) o moderador pode gerar outro token.

FCS-18 – Fluxo da Camada de Sessão

No passo 2 do fluxo-base, a camada de sessão realiza as seguintes ações:

1. A camada de sessão recebe o pedido da aplicação s_token_please, monta PDU com o seguinte formato: (id_sessionCSMu, id_service).
2. Envia o pedido (através da camada de transporte) ao moderador e aguarda resposta.

FCS-19 – Fluxo da Camada de Sessão

No passo 3 do fluxo-base, a camada de sessão realiza as seguintes ações:

1. Desmonta PDU de sessão (id_sessionCSMu, id_service).
2. Verifica se o IP do participante está cadastrado na tabela de sessões CSMu.
3. Verifica qual participante é o atual detentor do token.
4. Sinaliza aplicação.

FCS-20 – Fluxo da Camada de Sessão

No passo 4 do fluxo-base, a camada de sessão realiza as seguintes ações:

A camada de sessão recebe a resposta da aplicação e executa:
Se pedido de token aceito e token está com outro participante então
Monta PDU (aceito) e envia para quem solicitou o token (RT5).
Monta PDU e envia para outro participante (transição RT6).
Senão
Fim se

FCS-21 – Fluxo da Camada de Sessão

No passo 5 do fluxo-base, a camada de sessão realiza as seguintes ações:

1 Chegou a resposta.
2 Desmonta PDU e sinaliza aplicação (se aceito ou rejeitado).

FCS-22 – Fluxo da Camada de Sessão

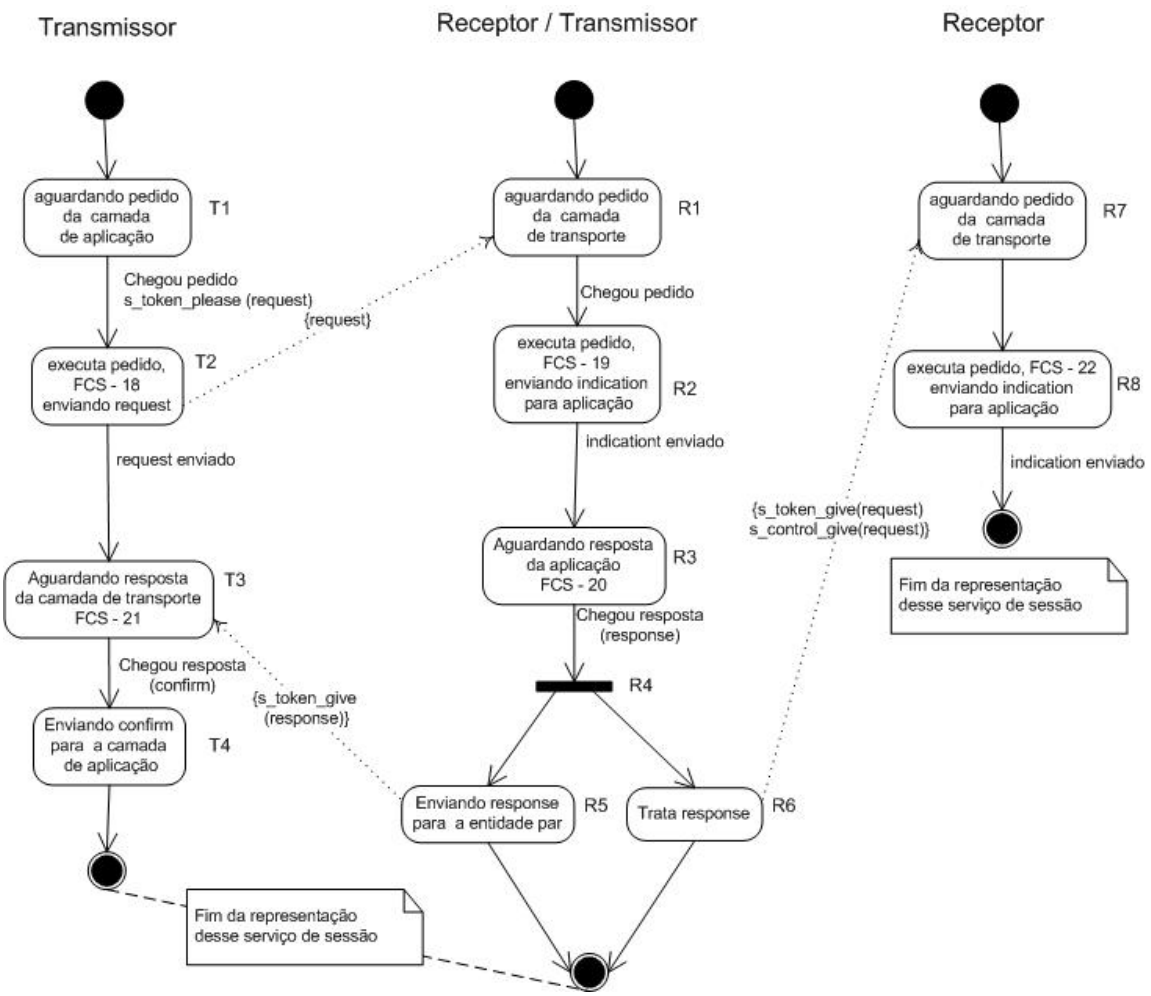
No passo 5 do fluxo-base, a camada de sessão realiza as seguintes ações:

1 Chegou a resposta.
2 Desmonta PDU e sinaliza aplicação.

INT-05: Interface da Camada de Sessão

PRIMITIVA	R	I	Rs	C	FUNÇÃO
Serviço Com Sessão Confirmado					
S_TOKEN_GIVE	•	•			Passagem de ficha de dados
S_TOKEN_PLEASE	•	•	•	•	Pedido de ficha
S_CONTROL_GIVE	•	•			Passagem de todas as fichas

ME 004 : Maquina de Estados



Estados	Descrição	Evento de ativação	Ação
Ativo	A conexão de sessão existe	Ao receber uma das primitivas: s_connect_M (accept). response s_connect_M (accept). confirm	Executa funcionalidades de sessão CSMu. Aguarda pedido finalização de conexão de sessão.
Executa Pedido	Executado pedido de gerência de diálogo	s_token_give s_token_please s_control_give	s_token_please : monta PDU e envia dados e aguarda resposta. s_token_give e s_control_give: monta PDU e envia dados.

A.7 Estabelecimento de CSMu

Descrição: Este caso de uso descreve o pedido de estabelecimento de conexão de sessão CSMu.

Fonte e/ou documentos relacionados:

- Norma X225_1_NormaSessãoOSI_ITUT
- Norma X225_2_NormaSessãoOSI_ITUT
- Capítulo 4.

Pendências: N/A

Ator ativo: Aplicação multimídia.

Outros Atores: N/A

Pré-Condições:

- N/A

Fluxo-base:

1. A aplicação pede o estabelecimento de conexão de sessão através da primitiva s_connect_m.request (INT-01).
2. A camada de sessão executa recebe o pedido de estabelecimento de conexão de sessão (FCS-01) e envia request para a entidade par; segue o diagrama de estados (MA-01).
3. A entidade par recebe o pedido da camada de transporte e executa os fluxos (FCS-02) e (FCS-03), monta PDU e envia a resposta.
4. Em FCS-04 o pedido de conexão foi aceito. Fim no caso de uso.
5. Em FCS-04 o pedido de conexão foi rejeitado. Fim no caso de uso.

Pós-Condição (Fluxo-base):

- A execução do passo 4 exclui a execução do passo 5 e vice-versa.
- Se a conexão foi aceita como resultado têm-se um identificador de sessão.

FCS-1 – Fluxo da Camada de Sessão

No passo 2 do fluxo-base, a camada de sessão realiza as seguintes as ações:

1. Armazena os dados enviados pela aplicação, através da primitiva s_connect_m, que contém o seguinte formato: (id_service, id_classes, id_categoria, id_instancia, endereço do buffer de sincronização, endereço do buffer de player).
2. Abre um canal de transporte confiável, preenche a PDU de pedido de conexão com os seguintes campos: (IP local, porta TCP local, IP destino, porta TCP destino, id_service, id_classes, id_categoria, id_instancia)
3. Envia o request para a entidade par.
4. Aguarda resposta da entidade par (servidor).

Pós-Condição (FCS-1)

- A aplicação ou a camada de sessão pode cancelar a operação encerrando esse caso de uso.

FCS-2 – Fluxo da Camada de Sessão

Pré-Condição (FCS-02): A entidade par está aceitando pedidos de conexão de sessão através de uma porta TCP. Existe um socket aberto no estado *listen*.

No passo 3 do fluxo-base, a camada de sessão realiza as seguintes ações:

1. Recebe PDU de pedido de estabelecimento de sessão. Através da primitiva id_service se identifica que é um pedido de sessão *multicast* (CSMu).
2. Identifica a instância da aplicação (PID do processo no Unix). Verifica na tabela de sessões CSMu, se existe um id_session_CSMu associado à essa aplicação.
    i. Se existe id_sessionCSMu então => armazena este id_sessionCSMU; vai para o passo 5 (FCS-2).
    ii. Senão => vá para o passo 3 (FCS-2).
3. Calcula identificador de sessão.
4. Valida id_session (num mesmo servidor, os id_session (CSMu) devem ter valores diferentes).
5. Verifica as classes de serviço (id_classes) que a camada pode atender.
6. Se o campo id_categoria estiver preenchido, preenche campo na tabela de sessões, indicando a categoria do tráfego.
7. Notifica a aplicação sobre os dados recebidos.
8. Aguarda resposta da aplicação (via SAP – Ponto de Acesso de Serviço de Sessão).

FCS-3 – Fluxo da Camada de Sessão

No passo 3 do fluxo-base, a camada de sessão realiza as seguintes ações:

1. Recebe resposta da camada de aplicação (via SAP).
2. Se conexão aceita então armazena dados na tabela de conexões CSMu, onde o formato do registro:
   Formato do registro:
   id_sessionCSMu,
   id_classes,
   id_categoria,
   id_instância,
   tempo_de_atividade,
   tabela_de_receptores:
   (receptor (lista de participantes), quantidade_max, &buffer_de_sincronização).

3. Monta PDU com a resposta e envia para a entidade par, contendo os seguintes dados: (IP local, porta TCP local, IP destino, porta TCP destino, id_service, id_classes, id_sessionCSMu, tempo de atividade).

FCS-4 – Fluxo da Camada de Sessão

No passo 4 do fluxo-base, a camada de sessão realiza as seguintes ações:

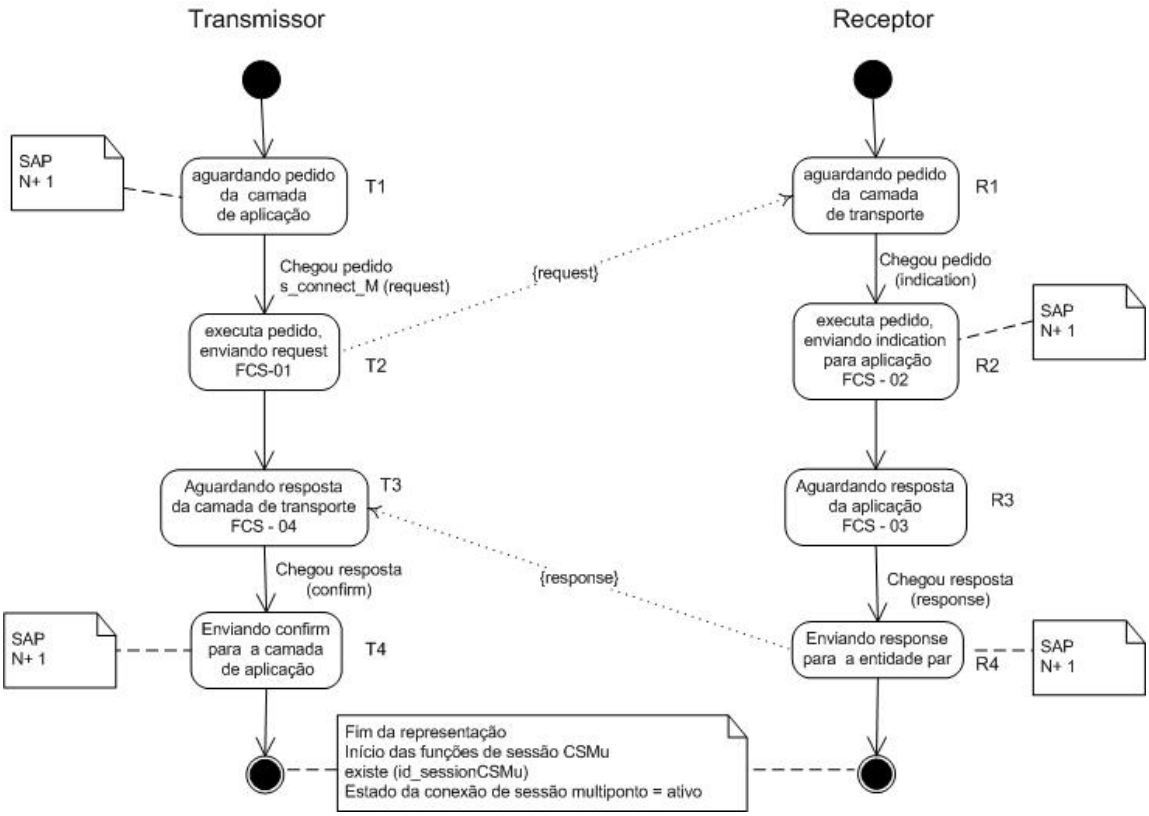
1 Recebe resposta da entidade par.
2 Desmonta PDU.
3 Verifica validade do id_session. Se campo nulo notifica aplicação (s_aut_report) => passo 6.
4 Se a conexão foi aceita então armazena registro na tabela de conexões *multicast*.
Formato do registro:
id_sessionCSMu,
id_classes,
id_categoria,
id_instância,
tempo_de_atividade,
tabela_de_receptores (quantidade_max) => (cliente, &buffer_de_sincronização).

5 Se a conexão NÃO foi aceita por um dos motivos abaixo:
a) recebeu um reject da entidade par -> sinaliza a aplicação através de um s_p_exception_report.
b) o temporizador expirou e não veio resposta -> sinaliza a aplicação através de um s_u_exception_report.
c) Se o servidor não oferece um ou mais dos serviços requisitados -> sinaliza a aplicação através de um s_message_report.
6 Notifica a aplicação: aceita (4), ou rejeitada(5).
7 Termina caso de uso.

INT-01: Interface do Componente Controle de Sessão

PRIMITIVA	R	I	Rs	C	FUNÇÃO
Serviço Orientado a Conexão					
S_CONNECT	•	•	•	•	Estabelecimento de conexão
S_CONNECT_M	•	•	•	•	**Estabelecimento de conexão *multicast***
S_RELEASE	•	•	•	•	Liberação negociada de conexão
S_U_ABORT	•	•			Liberação abrupta (usuário)
S_P_ABORT		•			Liberação abrupta (fornecedor)
S_U_EXCEPTION_REPORT	•	•			**Relatório de anomalia (usuário)**
S_P_EXCEPTION_REPORT		•			**Relatório de anomalia (fornecedor)**
S_ MESSAGE_REPORT	•	•			**Mensagem imprópria**
S_ AUT_REPORT	•	•			**Falha na autenticação do id_session**

MA-01 : Maquina de Estados



Estados	Descrição	Evento de ativação (condição)	Ação
Ocioso	Estado inicial da camada de sessão, quando não existe sessão CSMu.	Ao ligar o sistema Ao receber uma das primitivas: s_connect_m (reject). response Ao receber um pedido de liberação de conexão de sessão confirmado (MA-02)	Aguardando pedido de estabelecimento de conexões de sessão CSMu.
Executa Pedido	Executando pedidos de conexão de sessão	s_connect_m(request), s_connect_m(indication)	Lado transmissor: FCS-01 Lado receptor: FCS-02
Aguarda Resposta	Aguarda resposta do pedido de estabelecimento de sessão	Recebe pedido de serviço confirmado	Lado receptor: FCS-03 Lado transmissor: FCS-04
Ativo	Existe uma ou mais conexões de sessão ativas	s_connect_m (accept). confirm	Estabelece conexão, armazena dados da CSMu, executa demais funcionalidades da camada de sessão.

A.8 Mecanismo de Player Local

Descrição: Este caso de uso descreve o mecanismo de player local da camada de sessão.

Fonte e/ou documentos relacionados:

- Capítulos 3 e 4.

Pendências: N/A

Ator ativo: Aplicação multimídia.

Outros Atores: N/A

Pré-Condições:

- Existe uma conexão de sessão estabelecida (id_session).
- A esse id_session existem associados duas áreas de armazenamento definidas em espaço de usuário (&buffer_sync, e &buffer_player), que serão alimentadas pelos serviços da camada de sessão.
- O mecanismo de player local não utiliza comunicação cliente-servidor. Esse é um serviço oferecido pela camada de sessão apenas para a aplicação local. Trata-se de um segundo buffer (maior que o de sincronização) utilizado para armazenamento dos dados, que responde aos comandos gerados pela aplicação local.

Fluxo-base:

1 A aplicação, através da interface entre camadas, inicia a função de player (s_player_start) INT-06.
2 A camada de sessão recebe solicitação (FCS-30)
3 A camada de sessão alimenta o buffer de player com os dados do buffer de sincronização (FCS-31).
4 A camada de sessão responde à solicitação de avanço e retrocesso da aplicação FCS-32 e FCS-33.
5 A aplicação finaliza o buffer de player (s_player_end) (INT-06), FCS-31.

Pós-Condição (Fluxo-base):

- O início do buffer local equivale ao ponto inicial da transmissão dos dados.
- As funções de avanço e retrocesso são aplicadas a um buffer local e não mais a um servidor remoto.

FCS-30 – Fluxo da Camada de Sessão

No passo 2 do fluxo-base à camada de sessão realiza as seguintes ações:

1 A camada de sessão recebe o pedido da aplicação (s_player_start ou s_player_end), contendo o identificador de sessão (id_session)
2 A camada de sessão busca na tabela de sessões (&buffer_player).

FCS-31 – Fluxo da Camada de Sessão

No passo 3 do fluxo-base à camada de sessão realiza as seguintes ações:

1 Se s_player_start então:
Quando buffer_sync estiver cheio => direciona o os dados do buffer_sync para o buffer_player.
Se buffer_player estiver cheio => descarta dados mais antigos.

2 Se s_player_end então:
Quando buffer_sync estiver cheio => descarta dados;

Pós-Condição (FCS-31):

- A execução do passo 1 exclui a execução do passo 2 e vice-versa.

FCS-32 – Fluxo da Camada de Sessão

No passo 4 do fluxo-base à camada de sessão realiza as seguintes ações:

1 Implementa o mecanismo de gerência de buffer, seguindo as seguintes regras:

a) Cada ponto de sincronismo marcado pela aplicação equivale a um ponto de avanço/retrocesso no buffer de player;
b) Quando o buffer_player estiver cheio, ele descarta os dados mais antigos;
c) Caso dados sejam descartados, o início do buffer não marcará mais o início da transmissão. O ponto máximo da função de retrocesso é o início do buffer de player e o ponto máximo da função de avanço é o final do buffer de sincronização.
d) Os dados só serão buscados no servidor, se eles não estiverem armazenados em nenhum dos dois buffers;
e) Quanto maior o buffer de player, menor a latência da rede.

FCS-33 – Fluxo da Camada de Sessão

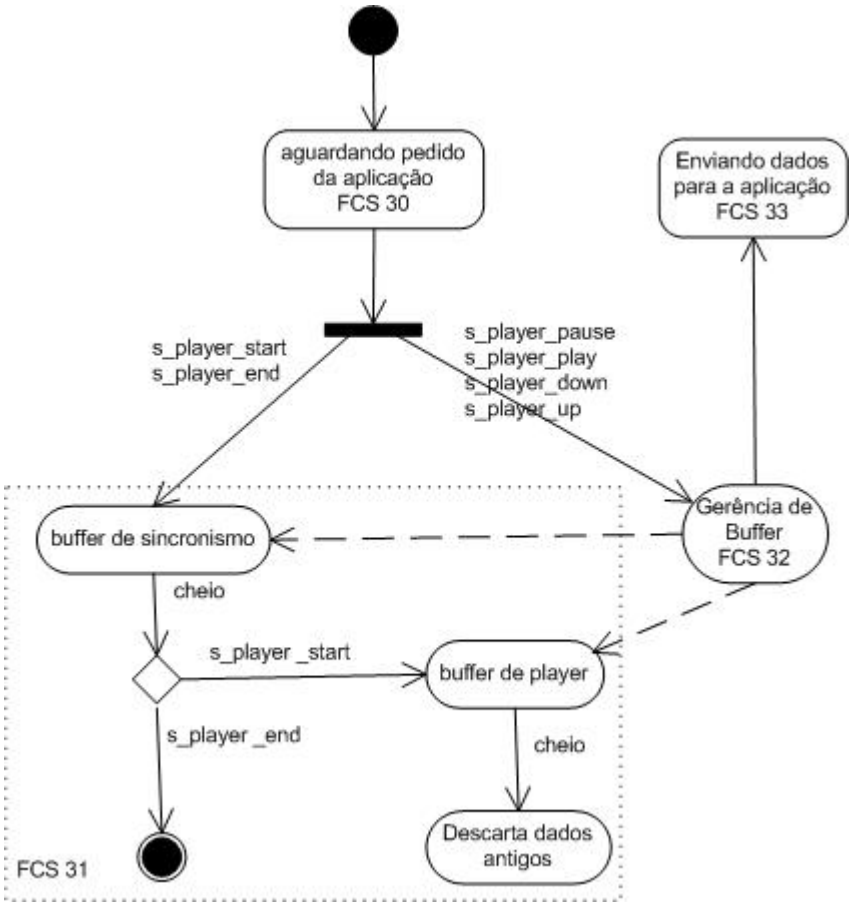
No passo 5 do fluxo-base à camada de sessão realiza as seguintes ação:

Sinaliza para a aplicação (posiciona o ponteiro e buffer) onde estão os dados que ela irá consumir.

INT-06 : Interface do Mecanismo de Player Local

PRIMITIVA	R	I	Rs	C	FUNÇÃO
Serviço Com Sessão Confirmado					
S_PLAYER_START	•			•	Início do mecanismo de player
S_PLAYER_END	•			•	Fim do mecanismo de player
S_PLAYER_PLAY	•			•	Aplicação sinaliza que está acessando os dados do buffer de player
S_PLAYER _PAUSE	•			•	Aplicação sinaliza que parou de acessar os dados do buffer de player
S_PLAYER_UP	•			•	Função de avanço
S_PLAYER_DOWN	•			•	Função de retrocesso

DA 05 : Diagrama de Estados



Estados	Descrição	Evento de ativação	Ação
Ativo	A conexão de sessão existe	Ao receber uma das primitivas: s_connect (accept). response s_connect(accept). confirm	Executa funcionalidades de sessão Aguarda pedido finalização de conexão de sessão
Executa Pedido	Executado pedido player local	s_player_start, s_player_end s_player_play, s_player_pause s_player_down,s_player_up	Inicializa e termina armazenamento de dados no buffer de player. Controla o buffer de player.

A.9 Mecanismo de Player Distribuído

Descrição: Este caso de uso descreve o mecanismo de player distribuído da camada de sessão.

Fonte e/ou documentos relacionados:
- Capítulos 3 e 4

Pendências: N/A

Ator ativo: Aplicação multimídia.

Outros Atores: N/A

Pré-Condições:

- Existe uma conexão de sessão *multicast* estabelecida (id_sessionCSMu).
- A função de player distribuído foi negociada no estabelecimento dessa sessão CSMu.
- Em cada participante da sessão CSMu existem associados duas áreas de armazenamento definidas em espaço de usuário (&buffer_sync, e &buffer_player), que serão utilizadas pelos serviços da camada de sessão.
- Os dados do buffer_player podem ser replicados para "n" participantes. O valor de "n" pode ser negociado no estabelecimento da sessão CSMu, e é utilizado pelo servidor para distribuir o fluxo de dados entre os participantes.

Fluxo-base:

1. O servidor de uma sessão CSMu informou para um participante seu vizinho mais próximo(FCS-34).
2. O participante pede para que seu vizinho encaminhe os dados de seu buffer de player (s_player_push_in) INT-06.
3. A camada de sessão envia pedido da entidade par (FCS-35).
4. A camada de sessão da entidade par recebe o pedido e executa-o (FCS-36).

Pós-Condição (Fluxo-base):

- O mecanismo de player distribuído replica os dados do buffer_player para vizinhos próximos.
- O player distribuído não implementa entre os participantes as funções do player local (avanço, retrocesso, ...)

FCS-34 – Fluxo da Camada de Sessão

No passo 1 do fluxo-base à camada de sessão realiza as seguintes ações:

1. O participante enviou ao servidor o pedido de participação na conferência CSMu (id_sessionCSMu, id_service);
2. O servidor utilizou o IP do participante e o endereço de rede para calcular o menor caminho;
3. O servidor identificou um cliente próximo do participante;
4. O servidor retornou uma PDU de informação (id_sessionCSMu, id_service, IP_cliente conectado ao servidor).

FCS-35 – Fluxo da Camada de Sessão

No passo 3 do fluxo-base à camada de sessão realiza as seguintes ações:

1 A camada de sessão recebe pedido s_player_push_in;
2 A camada de sessão monta PDU (id_sessionCSMu, id_service) e envia a entidade par.

FCS-36 – Fluxo da Camada de Sessão

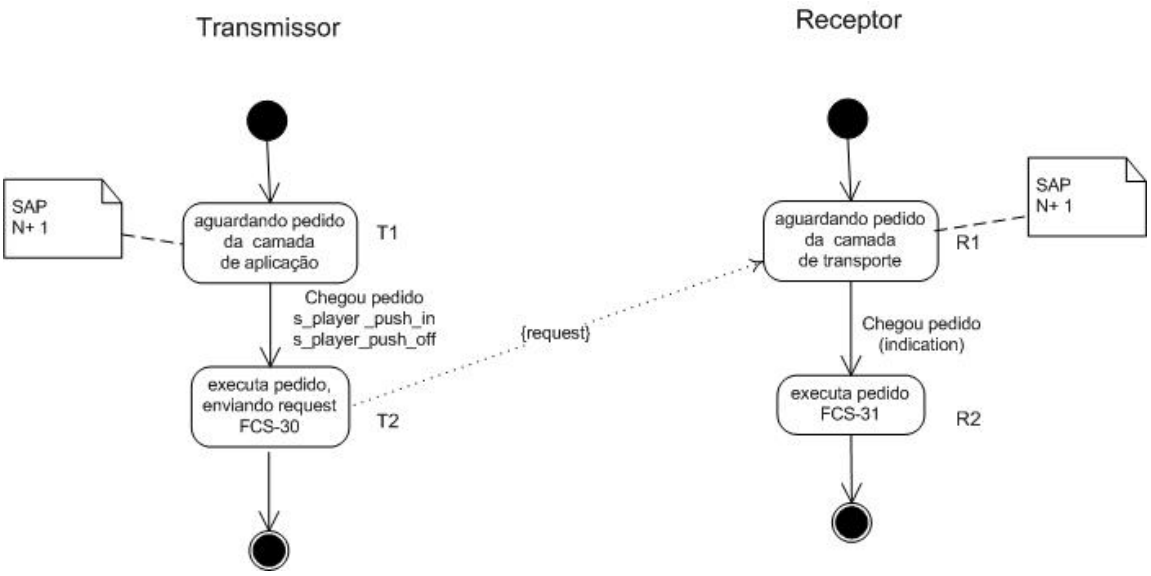
No passo 4 do fluxo-base à camada de sessão realiza as seguintes ações:

1 A camada de sessão recebe o pedido de utilização do buffer de player;
2 A camada de sessão adiciona cliente na tabela de sessões CSMu;
3 A camada de sessão monta PDU (id_sessionCSMU, idservice, IP_participante) e notifica o servidor da conexão CSMu, informando que o participante receberá os dados armazenados em seu buffer de player;
4 O servidor ao receber a PDU, deve atualizar o campo no registro da tabela de conexões *multicast*, adicionando o participante na lista de participantes:
tabela_de_receptores (receptor (lista de participantes), quantidade_max, &buffer_de_sincronização).
5 A camada de sessão abre um canal de comunicação com o participante. Este canal é responsável por replicar (enviar) os dados recebidos do servidor.

INT-06 : Interface do Mecanismo de Player Local

PRIMITIVA	R	I	Rs	C	FUNÇÃO
Serviço com Sessão Confirmado					
S_PLAYER_PUSH_IN	•	•			Pedido de envio dos dados do buffer de player
S_PLAYER_PUSH_OFF	•	•			Encerramento de envio dos dados do buffer

MA-06 : Máquina de Estados



Estados	Descrição	Evento de ativação	Ação
Ativo	A conexão de sessão existe	Ao receber uma das primitivas: s_connect (accept). response s_connect(accept). confirm	Executa funcionalidades de sessão Aguarda pedido finalização de conexão de sessão.
Executa Pedido	Executado pedido de player distribuído	s_player_push_in s_player_push_off	Inicializa transmissão dos dados do buffer de player; Termina transmissão dos dados do buffer de player;

APÊNDICE B – MODELO DE REFERÊNCIA RM-OSI

Para abordar a solução proposta, este trabalho considera alguns conceitos fundamentais que embasam a teoria de redes de computadores. Este apêndice se inicia com a discussão do modelo de referência *Reference Model for Open Systems Interconnection* (RM-OSI), apresentando suas características, funcionalidades e operações. Após apresentar uma visão geral do modelo, um enfoque mais detalhado é direcionado à sua quinta camada, conhecida como camada de sessão.

O objetivo é apresentar uma revisão bibliográfica da camada de sessão do RM-OSI de forma a permitir uma análise das funcionalidades ali propostas, e buscar em suas especificações definições que possam ser utilizadas no desenvolvimento do trabalho.

B.1 O modelo de referência RM-OSI

A *International Standards Organization* (ISO) foi uma das primeiras organizações que formalmente definiu uma maneira para conectar computadores. O modelo RM-OSI define o particionamento das funcionalidades da rede dentro de sete camadas, onde um ou mais protocolos implementam funcionalidades que estão associadas a uma camada específica. A Figura B.1 apresenta a arquitetura de rede OSI.

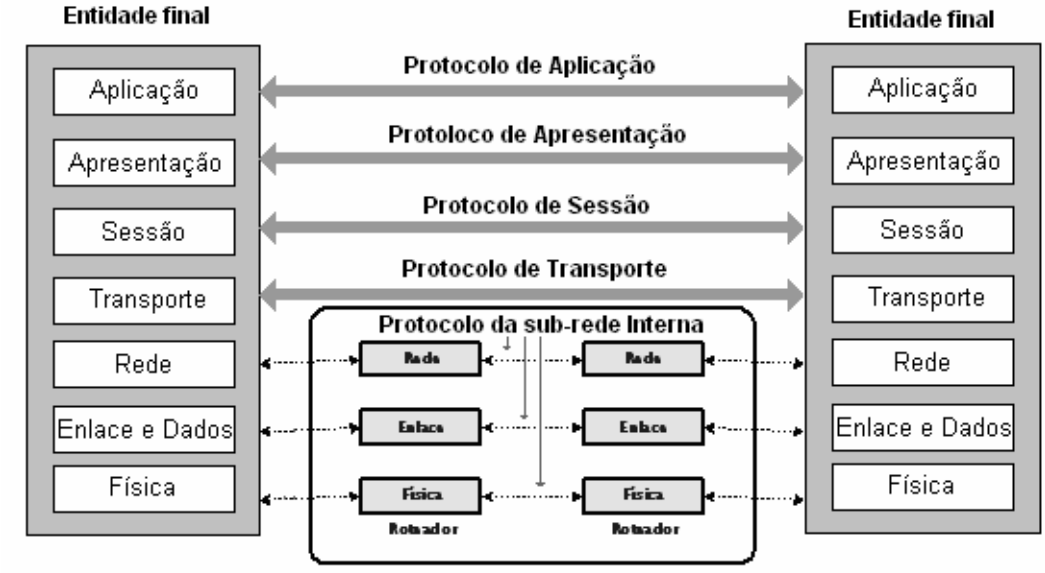


Figura B.1 – Arquitetura de rede OSI. Fonte: Adaptado de Tanenbaum (2003)

Esse conceito de divisão em camadas é extremamente importante em função da natureza complexa de uma rede. As camadas ajudam a reduzir essa complexidade simplificando o projeto de uma rede, pois tal abordagem possibilita que as funções e os serviços de uma determinada camada sejam completamente isolados das camadas adjacentes. Dessa forma, é possível alterar os recursos de uma camada sem modificar significativamente toda a arquitetura. Assim, à medida que novas tecnologias vão surgindo para uma camada específica, elas podem ser implementadas sem necessidade de alterar as demais camadas.

Os princípios de definição das sete camadas são os seguintes (TANENBAUM, 2003):

a) uma camada deve ser criada se houver necessidade de um nível de abstração no modelo;
b) cada camada possui suas funções próprias e bem definidas;
c) as funções de cada camada devem ser escolhidas segundo a definição dos protocolos padronizados internacionalmente;
d) as fronteiras entre cada camada foram definidas de modo a minimizar o fluxo de informações pelas interfaces; e
e) o número de camadas deve ser suficientemente grande para que funções distintas não precisem ser colocadas na mesma camada e suficientemente pequeno para que a arquitetura não se torne difícil de controlar.

As camadas do modelo RM-OSI, de cima para baixo, são: aplicação, apresentação, sessão, transporte, rede, enlace de dados e física. O Quadro B.1, a seguir, apresenta um resumo de suas respectivas funções (GALLO; HANCOCK, 2003).

CAMADA	FUNÇÕES
Aplicação (7)	Trata dos serviços e procedimentos relativos a aplicações de usuários. Compreende os protocolos de aplicações específicas tais como correio eletrônico (protocolo X.400) e transferência de arquivos (FTAM – *File Transfer, Access and Management*).
Apresentação (6)	Trata da codificação e formatação dos dados (inclui criptografia). Cuida da sintaxe e da semântica dos dados transmitidos. Recebe as mensagens da camada de aplicação, formata-as e transmite-as para a camada de sessão.
Sessão (5)	Controla comunicação entre os processos. Responsável pelas regras de diálogo, pela sincronização e armazenamento intermediário do fluxo de dados e pelo restabelecimento da conexão em caso de falha.
Transporte (4)	Trata do controle fim-a-fim, ou seja, a entrega dos dados sem erros. Aceita dados da camada de sessão. Se necessário, quebra-os em unidades menores, passa essas unidades para a camada de rede e garante que elas cheguem ao destino completas e corretas. No destino, reestrutura as unidades e devolve-as para a camada de sessão.

Rede (3)	Trata dos serviços de rotas para transferir dados pela rede. É responsável pelo controle de congestionamento da rede e pela transmissão de dados em redes heterogêneas. As mensagens formatadas são denominadas pacotes.
Enlace e Dados (2)	Responsável pela transferência de dados entre pontos de uma ligação física. Organiza os dados em quadros (enquadramento), trata da detecção dos erros e controle de fluxo. Resolve problemas relativos a quadros danificados, perdidos ou duplicados.
Física (1)	Responsável pela transmissão de bits através de uma ligação. Define características dos meios físicos, tais como tipo de cabo em uso e tipo de conector. Aceita quadros da camada de enlace e traduz os bits em sinais do meio físico.

Quadro B.1 – Funcionalidade das camadas

Cada camada é composta de Serviço, Interface e Protocolo. O Serviço define os tipos de serviços que uma camada é capaz de oferecer para a sua camada imediatamente superior, determinando o que a camada é capaz de executar, sem se preocupar em como fazer. A Interface informa o funcionamento da camada, como ela pode ser acessada e quais são os parâmetros e resultados que serão esperados. O Protocolo detalha como fazer a implementação de um determinado serviço, definindo um conjunto de regras que detalham o significado das unidades de dados trocadas entre entidades pares de uma camada. Uma camada pode utilizar quantos protocolos ela quiser na implementação de um determinado serviço (TANENBAUM, 2003).

A camada N utiliza os serviços da camada N-1 e oferece serviços à camada N+1. Seguindo essa lógica, o RM-OSI define pontos de acesso aos (serviços) *Service Access Point* (SAP), que são os lugares onde a camada N+1 pode acessar os serviços oferecidos pela camada N.

A Figura B.2 demonstra a interação entre as camadas do modelo RM-OSI e seus respectivos parâmetros de serviço. Uma transferência de dados se inicia quando o usuário da camada superior, através da interface da camada, utiliza uma primitiva de serviço. Isso resulta na configuração de uma unidade de dados do protocolo (*Protocol Data Unit* - PDU) gerada, dentro da camada, pelo protocolo local, que utilizará os serviços da camada inferior com a finalidade de repassar a PDU para a entidade par correspondente do sistema remoto. Ao passar para a camada imediatamente inferior, a PDU é acrescida de uma informação adicional de controle de protocolo, denominada *Protocol Control Information* (PCI). A camada de enlace é a única a acrescentar um PCI próprio, diferente dos demais, onde os dados são codificados e transmitidos para o sistema remoto. No sistema remoto, cada camada interpreta o seu PCI, retira-o e repassa a PDU para a camada imediatamente superior (HALSALL, 1997).

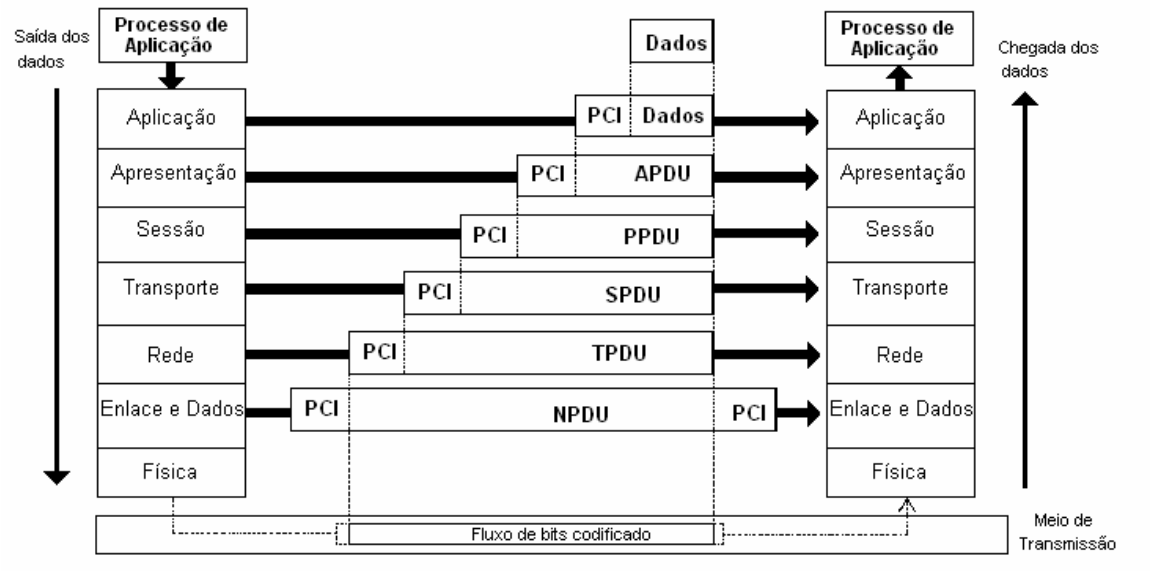


Figura B.2 – Múltiplas camadas. Fonte: Adaptado de Halsall (1997)

B.1.1 Primitivas de Serviço OSI

Os serviços de usuário fornecidos por uma camada são especificados por um conjunto de primitivas de serviço, que são divididas em quatro classes: *request*, *indication*, *response* e *confirm* (TANENBAUM, 1996). O Quadro B.2, abaixo, mostra o significado de cada uma dessas classes.

CLASSES	SIGNIFICADO
Request	Pedido enviado por uma entidade que solicita um serviço
Indication	Através dela, a entidade par é informada de uma solicitação de serviço.
Response	A entidade par responde ao pedido de serviço
Confirm	A entidade solicitante é informada do resultado do serviço

Quadro B.2 – Primitivas de serviço

Os serviços associados a uma determinada camada podem ser de dois tipos: confirmado ou não confirmado. O serviço confirmado é composto de *request*, *indication*, *response* e *confirm*. O serviço não confirmado é composto apenas de primitivas do tipo *request* e *indication*. A Figura B.3 ilustra a diferença entre os dois tipos de serviços.

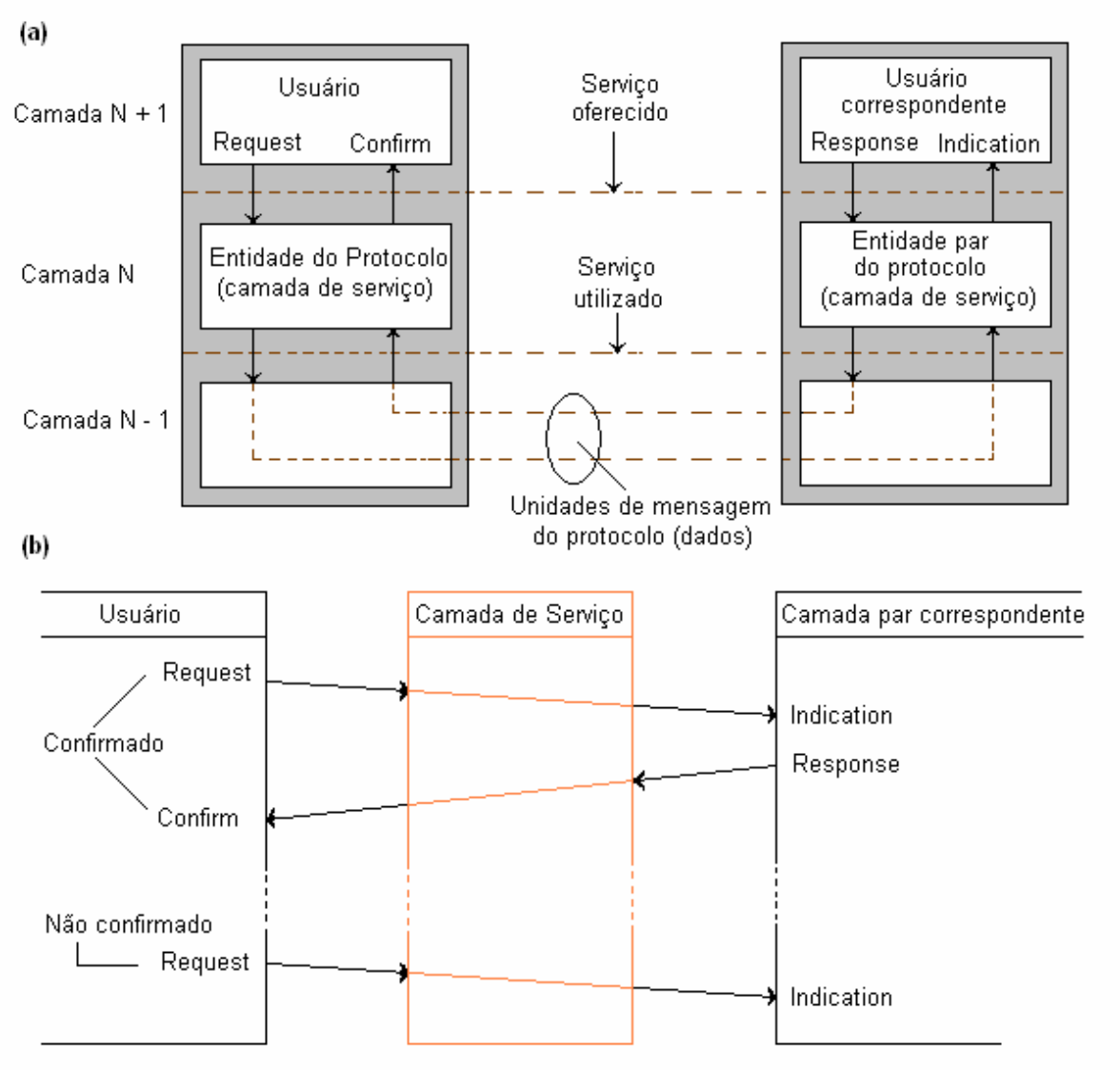


Figura B.3 – Primitivas de serviço: (a) representação de espaço; (b) representação de seqüência. Fonte: Adaptado de Halsall (1997)

B.2 A Camada de Sessão

O modelo de referência RM-OSI, diferentemente do TCP/IP, prevê uma camada exclusiva para prestação dos serviços de estabelecimento, gerenciamento e término de sessões. Esse tipo de abordagem facilita o desenvolvimento de aplicações, uma vez que elas não precisam implementar esses serviços.

De acordo com a recomendação X.215 do ITU-T (1995, p.16):

> "[...]o serviço de sessão provê os meios necessários para troca organizada e sincronizada de dados entre usuários co-operantes da camada de sessão."

Uma entidade de aplicação, através de sua relação com uma entidade de apresentação, pode utilizar os seguintes serviços da camada de sessão (HALSALL, 1997):

a) estabelecimento de um caminho lógico de comunicação (uma conexão de sessão) com outra entidade de aplicação;
b) estabelecimento de pontos de sincronismo utilizados durante a transferência de dados, que permitem que, na ocorrência de uma falha durante uma transação de dados, o diálogo possa ser reiniciado a partir do último ponto de sincronismo;
c) interrupção e retomada do diálogo a partir de um ponto prédefinido; e
d) relatório de exceções, que podem ocorrer desde a base da rede, sendo sinalizadas à camada de aplicação através da camada de sessão.

Entidades de aplicação podem utilizar tanto o modo de comunicação com conexão quanto o modo sem conexão. Esses dois modos são suportados pelos protocolos de apresentação e de sessão. No modo com conexão, uma conexão de sessão é mapeada sobre uma ou mais conexões de transporte (não necessariamente simultâneas), ou, ainda, duas ou mais conexões de sessão podem ser mapeadas sobre uma única conexão de transporte, conforme visualizado na Figura B.4 (TANENBAUM, 1991).

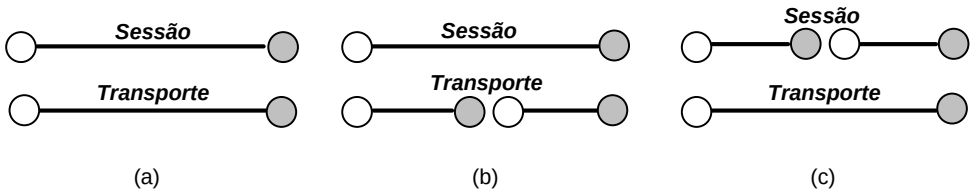


Figura B.4 – Diferentes relações entre conexão de sessão e de transporte:(a) correspondência um a um; (b) várias conexões de transporte para uma única *sessão*;(c) uma conexão de transporte para várias *sessões*. Fonte: Adaptado de Tanenbaum (1991).

Neste modo, o modelo prevê estabelecimento, transferência de dados, gerenciamento (envolvendo as funcionalidades de controle de fluxo, sincronização de diálogos, gerenciamento de tokens e atividade) e término da sessão. No modo sem conexão, a função da camada de sessão é apenas mapear os endereços de transporte em endereços de sessão.

B.2.1 Gerência de Diálogo

Segundo o RM-OSI, todas as conexões estabelecidas são, em princípio, do tipo full-duplex, ou seja, a transferência de dados pode ser realizada nos dois sentidos, simultaneamente. Porém, existem aplicações nas quais a possibilidade de operação *half-*

duplex pode ser mais interessante. Nesses casos, a comunicação *half-duplex* deve ser negociada no momento do estabelecimento de uma conexão de sessão.

Para que duas entidades de sessão possam gerenciar o diálogo sobre uma conexão de sessão preestabelecida, foi definido um conjunto de serviços para a troca de permissões (tokens). Quando um token é atribuído a um usuário do serviço de sessão (*Session Service* – SS), é permitido que este faça uso de certos serviços com exclusividade. O conjunto de tokens definido é composto de token de dados, token de encerramento de conexão, token de sincronização secundária (menor) e token de sincronização principal (maior)/atividade.

Quando o modo *half-duplex* é usado, apenas o proprietário do token pode enviar dados. Então, quando a entidade par desejar enviar dados, ela deve primeiro pedir a posse do token. Somente após obter a posse do token é que os dados poderão ser enviados.

B.2.2 Sincronização

Quando dois usuários de um serviço de sessão enviam grande quantidade de dados, estes precisam ser estruturados dentro de um número de unidades identificáveis para que, se houver falha na comunicação, apenas aqueles transferidos mais recentemente sejam afetados. Para permitir que um usuário execute essa função, pontos de sincronização são inseridos dentro de blocos seqüenciais antes da transmissão de dados. Cada ponto de sincronização é identificado por um número serial, que é mantido pelo protocolo da entidade de sessão. Dois tipos de pontos de sincronização são fornecidos (HALSALL, 1997):

a) ponto de sincronismo maior: está associado a uma unidade de dados (diálogo) completa; e
b) ponto de sincronismo menor: está associado com partes de uma unidade de dados (diálogo).

Unidades de diálogo são intervalos de comunicação, delimitados pelos pontos de sincronização maior, que independem das unidades de diálogo anteriores e posteriores, e que podem ser mapeadas por uma função específica da aplicação.

Os tokens de sincronização menor e maior/atividade estão associados ao processo de sincronização, que pode ser usado durante uma sessão, definindo o fim de uma unidade de diálogo e o início de outra.

Os pontos de sincronização maior devem ser confirmados explicitamente pelas entidades comunicantes, enquanto os pontos de sincronização menor podem ou não ser confirmados. O

uso de pontos de sincronização menor durante uma unidade de diálogo é opcional e serve para garantir que os dados enviados anteriormente a esse ponto não necessitem ser retransmitidos no caso de uma ressincronização, o que aumenta a flexibilidade na recuperação de falhas.

A Figura B.5 detalha os conceitos de sincronização, unidade de diálogo e atividade, descritos no próximo item.

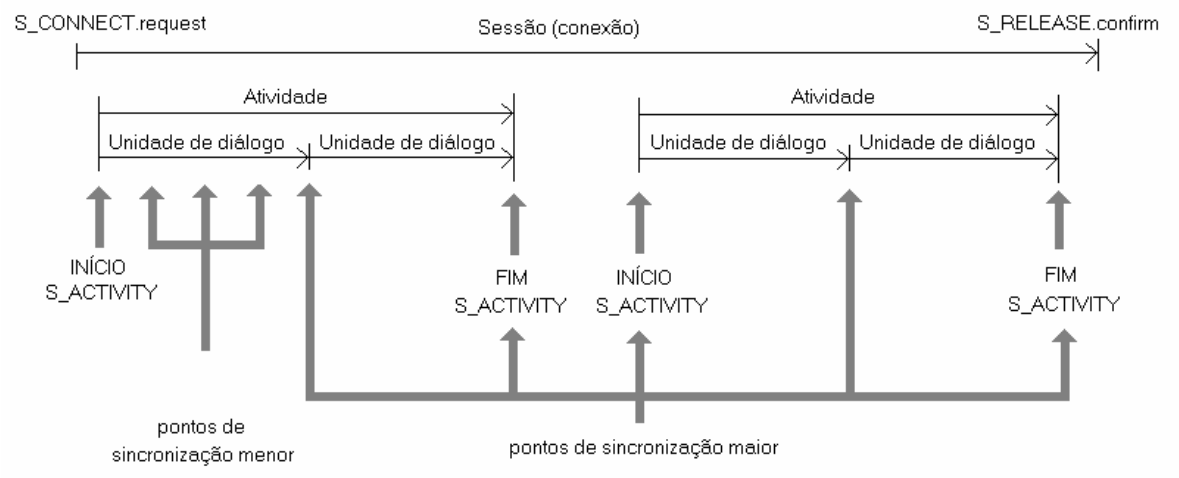


Figura B.5 – Detalhes da camada de sessão:conceitos de sincronização, unidade de diálogo e atividade. Fonte: Adaptado de Halsall (1997)

B.2.3 Gerência de Atividade

Atividades são unidades lógicas de um trabalho, constituídas por uma ou mais unidades de diálogo (Figura B.5). O conceito de atividade permite a dois usuários de um serviço de sessão distinguirem entre as partes lógicas do trabalho associado à sessão. Uma sessão pode transmitir várias atividades consecutivas, porém, num dado instante, somente uma atividade pode estar em progresso em uma conexão de sessão. O modelo também prevê a interrupção de uma atividade (para transmissão de uma atividade mais importante, ou suspensão de uma atividade muito longa) que pode ser retomada posteriormente na mesma conexão de sessão ou em outra conexão de sessão diferente. Essa retomada da atividade é realizada a partir de pontos de sincronização, e cabe ao usuário do serviço de sessão o armazenamento das informações necessárias para o reinício da atividade.

B.2.4 Unidades Funcionais da Camada de Sessão

Os serviços oferecidos pela camada de sessão são agrupados dentro de unidades funcionais. Este agrupamento lógico permite que dois usuários negociem, precisamente, quais serviços eles utilizarão. Com exceção da unidade funcional Kernel, todas as outras podem ser negociadas no momento do estabelecimento de uma conexão de sessão. O Quadro B.3 apresenta a descrição das sete unidades funcionais.

UNIDADE FUNCIONAL	DESCRIÇÃO DOS SERVIÇOS
Kernel (não negociável)	Fornece as funções básicas de estabelecimento, encerramento de conexão e transferência de dados nos dois sentidos (bidirecional).
Encerramento negociável	Fornece um serviço de liberação negociado. Fazem parte desta unidade as primitivas de serviço *give-token* e *please-token.*
Half Duplex	Fornece um envio alternativo (unidirecional) de dados. Fazem parte as primitivas de serviço *give-token* e *please-token.*
Sincronização	Fornece sincronização e re-sincronização durante a conexão de sessão.
Gerência de Atividade	Fornece identificação, inicialização, finalização, suspensão e reinicialização das atividades. Fazem parte desta unidade as primitivas: *give-token*, *please- token* e *give-control.*
Relatórios de Anomalias	Fornecem relatórios de anomalias (notificação de exceção pelo provedor e notificação de exceção pelo usuário) durante uma conexão de sessão.

Quadro B.3 – Unidades funcionais do serviço de sessão

De acordo com Halsall (1997), para prevenir que o usuário do serviço de sessão tenha que especificar cada unidade funcional no momento do estabelecimento de uma conexão de sessão, é necessário definir um número de subconjuntos que englobe diferentes combinações dessas unidades. Isso pode ser visualizado na Figura B.6, que apresenta os seguintes subconjuntos:

a) subconjunto básico combinado (SBC) – inclui as unidades funcionais Kernel e *half-duplex*;
b) subconjunto básico sincronizado (SBS) – inclui as unidades funcionais de sincronização; e
c) subconjunto básico de atividade (SBA) – inclui o gerenciamento de atividade e unidades de relatórios de sessão.

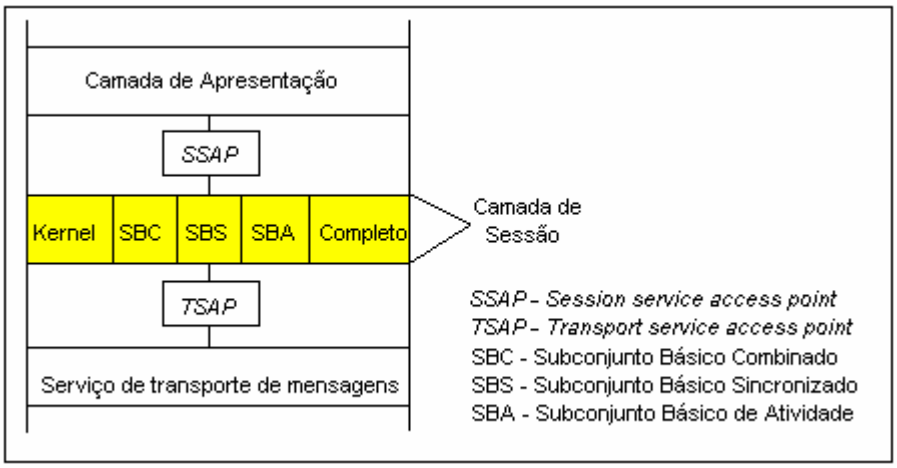


Figura B.6 – Detalhes da camada de sessão: subconjuntos do protocolo. Fonte Adaptado de Halsall (1997)

Primitivas de serviço de usuário são utilizadas para implementar todas essas funcionalidades. Por exemplo, para implementar um subconjunto básico combinado são utilizadas as seguintes primitivas: s_connect, s_data, s_token_please, s_token_give, s_release, s_u_abort e s_p_abort, descritas no próximo item.

A cada serviço oferecido pela camada de sessão são associados parâmetros. Por exemplo, em uma primitiva s_connect, dois usuários de um serviço de sessão negociam quais tipos de serviços (unidades funcionais) serão utilizados durante a conexão de sessão. Outro exemplo seria com relação aos parâmetros associados com duas primitivas de token, que incluiriam no processo de negociação o tipo de token (dados, liberação negociada de conexão e outros).

B.2.5 Primitivas de Serviço de Sessão

O quadro abaixo apresenta uma lista de primitivas de serviço oferecidas pela camada de sessão, organizadas em sete grupos: estabelecimento de conexão, liberação de conexão, transferência de dados, gerenciamento de *tokens*, sincronização, controle de atividades e relatório de anomalias.

PRIMITIVA	R	I	Rs	C	FUNÇÃO
ORIENTADO CONEXÃO					
S_CONNECT	•	•	•	•	Estabelecimento de conexão
S_RELEASE	•	•	•	•	Liberação negociada de conexão
S_U_ABORT	•	•			Liberação abrupta (usuário)
S_P_ABORT		•			Liberação abrupta (fornecedor)
S_DATA	•	•			Transferência de dados normais
S_EXPEDITED_DATA	•	•			Transferência de dados urgentes
S_TYPED_DATA	•	•			Transferência de dados classificados
S_CAPABILITY_DATA	•	•	•	•	Transferência de dados de capacidade

S_TOKEN_GIVE	•	•			Passagem de ficha de dados
S_TOKEN_PLEASE	•	•			Pedido de ficha
S_CONTROL_GIVE	•	•			Passagem de todas as fichas
S_SYNC_MAJOR	•	•	•	•	Inserção de ponto de sincronização máximo
S_SYNC_MINOR	•	•	•	•	Inserção de ponto de sincronização mínimo
S_RESYNCHRONIZE	•	•	•	•	Pedido de ressincronização
S_ACTIVITY_START	•	•			Início de uma atividade
S_ACTIVITY_END	•	•	•	•	Fim de uma atividade
S_ACTIVITY_DISCARD	•	•	•	•	Abandono de uma atividade
S_ACTIVITY_INTERRUPT	•	•	•	•	Interrupção de uma atividade
S_ACTIVITY_RESUME	•	•			Retomada de uma atividade
S_U_EXCEPTION_REPORT	•	•			Relatório de anomalia (usuário)
S_P_EXCEPTION_REPORT		•			Relatório de anomalia (fornecedor)
SEM CONEXÃO					
S_UNITDATA	•	•			Transferência de dados (sem conexão)

Quadro B.4 – Primitivas de serviços da camada de sessão

B.2.6 Protocolo de Sessão

Uma unidade de dados do protocolo de sessão (SPDU – *Session Protocol Data Unit*), visualizada na Figura B.7, é composta de identificador de sessão (SI – *Session identifier*), indicador do tamanho (LI – *Length indicator*) em octetos, identificador de parâmetro (*Parameter identifier*), valor de parâmetro (*Parameter value*) e identificador de grupo de parâmetro (PGI – *Parameter group identifier*). Os dois primeiros campos, SI e LI, estão presentes em todas as SPDUs. Dependendo da funcionalidade da primitiva de serviço, que compõe uma determinada SPDU, são definidos diferentes parâmetros por meio dos campos PI, LI e PV. Parâmetros também podem ser agrupados; nesse caso, uma *string* de parâmetro é precedida por um identificador de grupo de parâmetro (PGI) e por um identificador de cálculo de tamanho de parâmetro (LI).

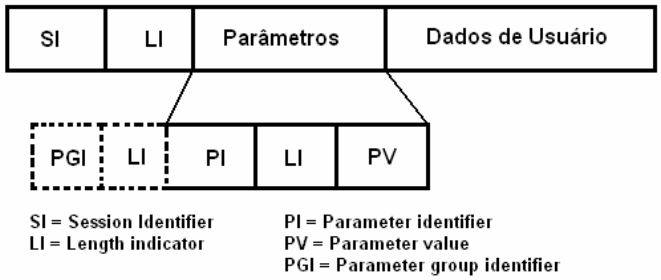


Figura B.7 – Formato da SPDU. Fonte: Adaptado de Halsall (1997)

Antes que qualquer SPDU possa ser enviada para a entidade par, é necessário que uma conexão de transporte já tenha sido estabelecida. Sobre essa conexão de transporte a SPDU

será enviada. A Figura B.8 ilustra como as SPDUs são utilizadas numa transferência de dados duplex e *half-duplex*, num encerramento abrupto e negociado entre duas entidades pares.

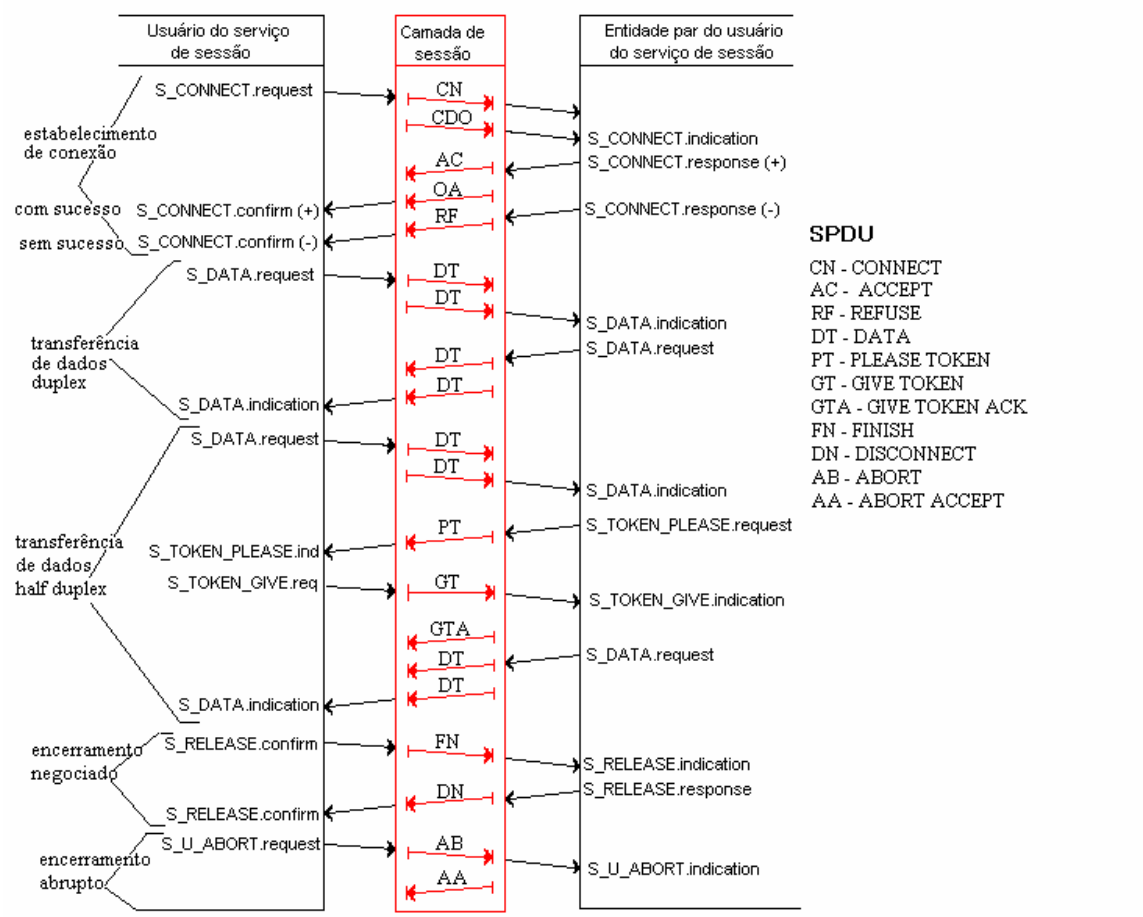


Figura B.8 – SPDUs (apenas do SBC). Fonte: Adaptado de Halsall (1997)

B.3 Considerações Finais

Este apêndice apresentou uma revisão sobre o modelo de referência RM-OSI, considerado por muitos um modelo conceitual, amplo e completo. No escopo deste trabalho propõe-se uma extensão à arquitetura da Internet. Nesse sentido, destaca-se a importância em se conhecerem as funcionalidades mapeadas pelo RM-OSI, a fim de aprofundar os conhecimentos sobre os serviços de sessão e identificar, no contexto atual da Internet, quais dessas funcionalidades estão implementadas e como foi realizada tal implementação.

O próximo apêndice descreve a pilha TCP/IP, sua estrutura e conceitos relacionados à implementação dos mecanismos de sockets, utilizados pelas aplicações no estabelecimento de uma comunicação entre cliente e servidor. A descrição dessa funcionalidade é necessária, já que o modelo proposto interage diretamente com as camadas de transporte e de aplicação da Internet, e tem como um de seus objetivos não interferir no funcionamento das aplicações atuais.

APÊNDICE C – A ESTRUTURA DE COMUNICAÇÃO DA ARQUITETURA INTERNET

A arquitetura Internet, também conhecida como arquitetura TCP/IP, originou de experiências desenvolvidas em uma rede de comutação de pacotes, denominada Arpanet. Tanto a Internet como a Arpanet foram fundadas pela Agência de Projetos de Pesquisas Avançadas (*Advanced Research Projects Agency* – ARPA) do Departamento de Defesa dos Estados Unidos da América (PETERSON; DAVIE, 2000).

Atualmente, quase toda a informação sobre a história da Internet e seus protocolos está disponível em *Request for Comments* (RFCs), que contêm anotações de trabalhos de pesquisa e desenvolvimento da comunidade, anotações de reuniões das organizações da Internet, descrições de protocolos e experimentos, e especificações de padrões. Os RFCs compreendem atualmente duas subséries: uma para informação (*for your information* –FYI); e outra para padrões (*standards documentos* – STDs) disponíveis *online* em http://www.rfceditor.org. Antes de os documentos serem publicados como RFC, eles ficam disponíveis à comunidade para serem avaliados. Esses rascunhos (*drafts*) são documentos temporários, válidos por seis meses, que podem ou não se tornar um padrão.

Basicamente, a arquitetura Internet é composta de quatro camadas: aplicação, transporte, inter-redes e host/rede. A Figura C.1(a) faz um comparativo entre as duas arquiteturas, e a Figura C.1(b) mostra uma representação alternativa.

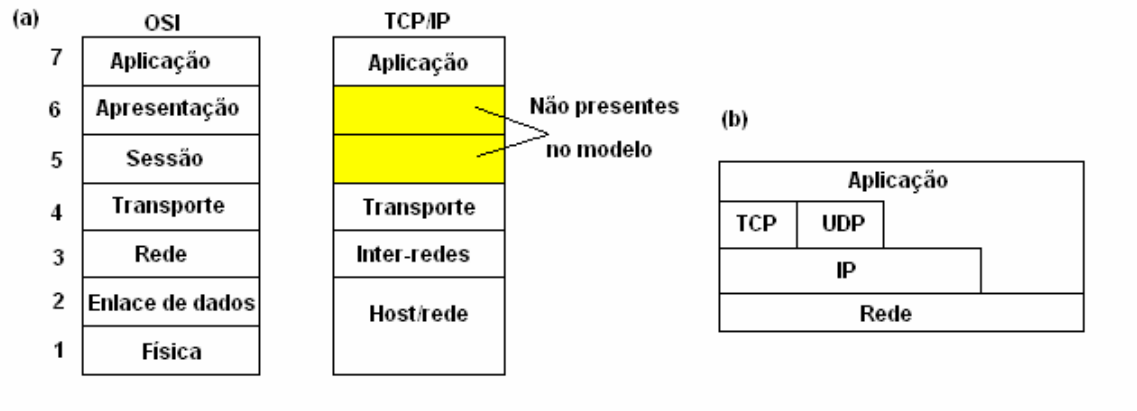


Figura C.1 – A arquitetura da Internet: (a) comparação com a arquitetura OSI. Fonte: Adaptado de Tanenbaum (2003); (b) representação alternativa. Fonte: Adaptado de Peterson e Davie (2000)

No nível mais baixo existe uma variedade de protocolos, implementados em combinação com o hardware, como, por exemplo, os protocolos do padrão Ethernet (IEEE 802.3). A segunda camada, denominada de Inter-redes, inclui o protocolo da Internet (IP), que é capaz de interconectar várias tecnologias de rede em uma única rede lógica. O IP é responsável pelo envio dos dados da sua origem até o seu destino, independentemente da localização, e junto com os protocolos *Address Resolution Protocol* (ARP), *Reverse Address Resolution Protocol* (RARP) e *Dynamic Host Configuration Protocol* (DHCP) oferece controle e resolução de endereços. Atualmente, duas versões do protocolo IP estão sendo utilizadas: a versão 4 (IPv4) e a versão 6 (IPv6).

A terceira camada, denominada camada de transporte, é composta de dois protocolos: o *Transport Control Protocol* (TCP) e o *User Datagram Protocol* (UDP). O TCP fornece um canal de envio de dados confiável, e o UDP fornece um canal de dados não confiável. Sobre a camada de transporte estão os protocolos de aplicação, tais como o *File Transfer Protocol* (FTP), o *Simple Mail Transfer Protocol* (SMTP), o *HyperText Transport Protocol* (HTTP) e uma infinidade de outros protocolos da Internet, necessários para o funcionamento das aplicações.

A flexibilidade da arquitetura Internet permite que uma aplicação acesse diretamente a tecnologia de rede, de acordo com a representação da Figura C.1(b). Como exemplo pode-se citar os equipamentos de videoconferência que utilizam a recomendação H.323 (definida pelo ITU-T). Fazem parte dessa recomendação os protocolos da camada de aplicação RTCP e RTP, definidos pelo IETF. Nesses equipamentos, as conexões de videoconferência podem ser feitas via arquitetura TCP/IP, utilizando o número IP como chamada, ou diretamente via conexão Rede Digital de Serviços Integrados[18] (RDSI).

A maioria das aplicações da Internet pode ser dividida em duas partes: cliente e servidor. Todo o fluxo de informação entre o cliente e o servidor passa pelo protocolo de transporte da Internet. Clientes e servidores são processos usuários, enquanto os protocolos TCP, UDP e IP são implementados no kernel do sistema operacional, como ilustrado na Figura C.2.

[18] Verificado pela autora em laboratório com os equipamentos da Polycom, que utilizam o protocolo H.323.

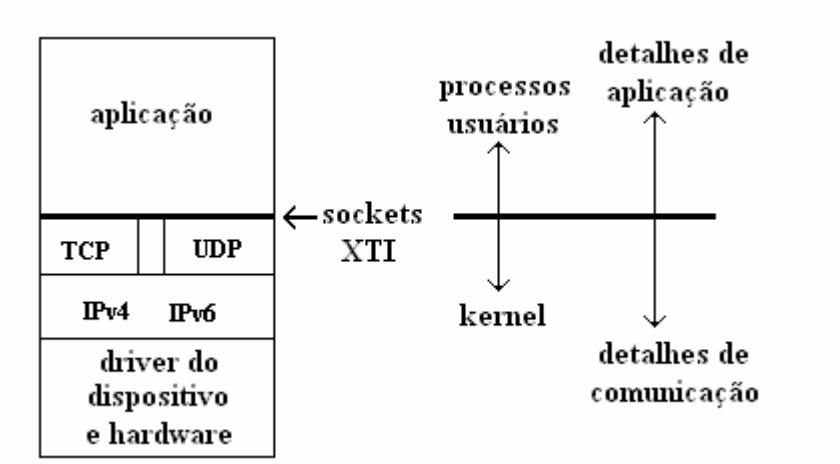


Figura C.2 – Conjunto de protocolos da Internet. Fonte: Adaptado de Stevens (1998)

Existe uma lacuna entre os protocolos de transporte TCP e UDP que indica a possibilidade de a aplicação acessar diretamente os protocolos de rede contornando os protocolos de transporte. De acordo com Stevens (1998), existem duas razões para a interface de sockets e a *X/Open Transport Interface*[19] (XTI) estarem localizadas dentro da camada de transporte. Primeiramente, as camadas de sessão, apresentação e aplicação do RM-OSI, cujas funcionalidades estão unificadas na camada de aplicação no modelo da Internet, quase não precisam conhecer os detalhes de comunicação, tais como envio de dados, espera pela confirmação, seqüenciamento, entrega desordenada, cálculo de *checksums* e outros problemas relacionados especificamente com a comunicação de dados. A segunda razão é que muitas das funcionalidades das três camadas superiores formam o que é chamado de processo usuário, enquanto as outras quatro camadas são normalmente concebidas como parte do kernel do sistema operacional. O sistema Unix, assim como muitos outros sistemas operacionais contemporâneos, faz uma separação entre processo usuário e kernel. Dessa forma, uma interface entre a camada de aplicação e as demais camadas constitui uma *Application Programming Interface* (API).

A seguir aborda-se a implementação da arquitetura Internet no sistema Unix, no que corresponde a comunicações entre processo usuário (camadas de sessão, apresentação e aplicação do RM-OSI) e ao kernel do sistema operacional (camadas de transporte e rede).

[19] XTI (*X/Open Transport Interface*) foi o nome dado em reconhecimento ao trabalho feito pelo *X/Open*, um grupo internacional de vendedores de computadores, entre eles a AT&T, que produziu seu próprio conjunto de padrões. XTI é efetivamente um conjunto extra da *Transport Layer Interface* (TLI).

C.1 Processos Usuários

Aplicações são processos usuários que enviam e recebem dados por meio de conexões de transporte (TCP e UDP). Esses processos usuários utilizam interfaces denominadas socket APIs para comunicar-se com o kernel do sistema operacional, onde estão implementadas as camadas de transporte, rede e enlace.

A Figura C.3 exemplifica o funcionamento de uma aplicação Web, em que um cliente (que possui um navegador Web) comunica-se com o servidor Web utilizando uma ou mais conexões TCP. A porta padrão de conexão com o servidor Web é a porta 80. Nesse exemplo, o protocolo de aplicação utilizado na comunicação entre cliente e servidor é o protocolo HTTP. Através de *hypertext links* um servidor Web pode apontar, por exemplo, para outro servidor Web ou para um servidor FTP. Para cada *hypertext link* existe uma diferente conexão TCP aberta entre o cliente e o servidor referenciado.

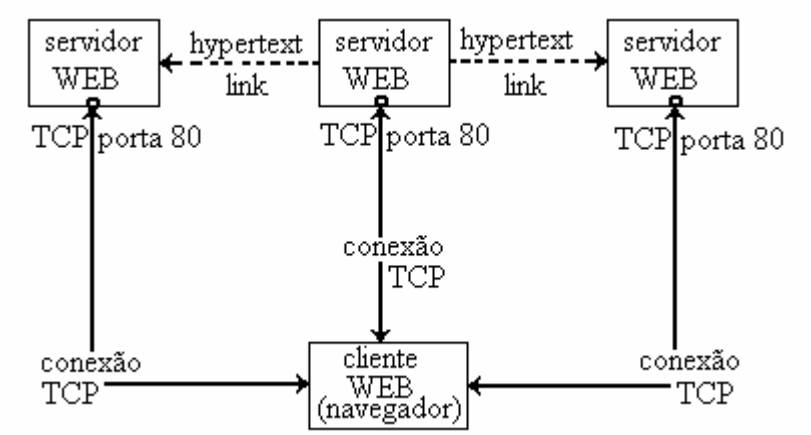


Figura C.3 – Organização de um cliente-servidor Web. Fonte: Adaptado de Stevens(1996)

A seção C.2, a seguir, apresenta com mais detalhes a estrutura de socket, utilizada pelas aplicações (processos usuários) no estabelecimento de um canal de comunicação.

C.2 Kernel – Camada de Transporte e Rede

O TCP/IP foi inicialmente implementado no sistema operacional Unix BSD 4.2, lançado em 1983. Seus desenvolvedores criaram uma API independente de protocolos, conhecida como *Berkeley Sockets* ou *BSD Sockets*, para possibilitar que as aplicações acessassem os

serviços dos protocolos da Internet (WRIGHT, 1995). A Figura C.4 detalha a organização da implementação do TCP/IP, conhecida como Net/3 (*BSD Networking Software, release 3.0*)[20].

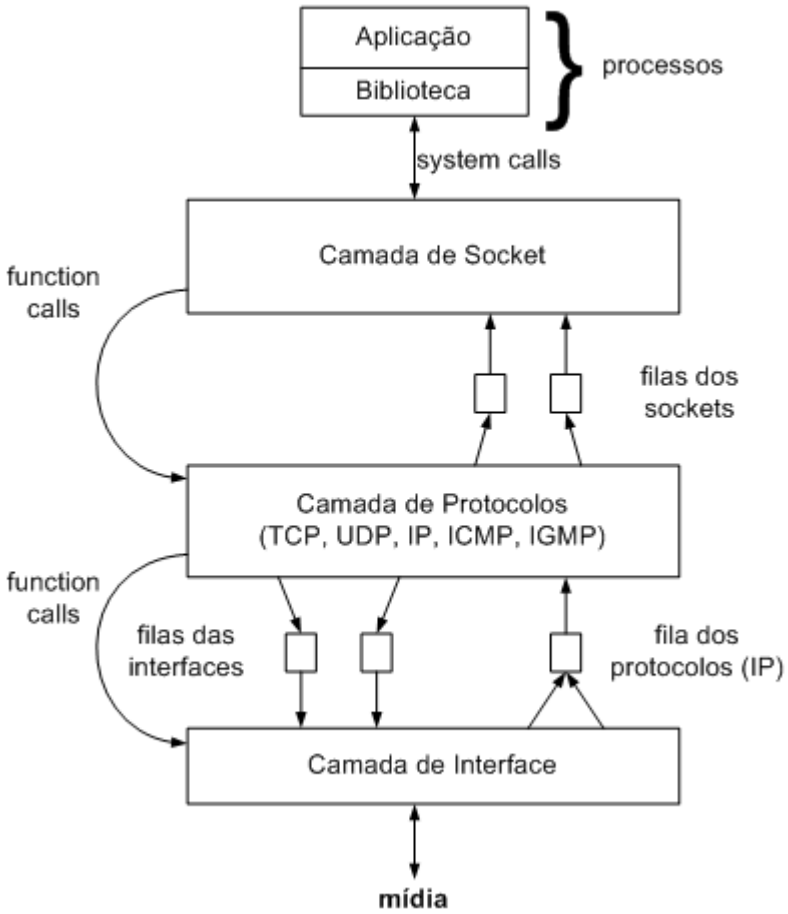


Figura C.4 – Comunicação entre as camadas na arquitetura do Net/3. Fonte: Adaptado de Wright (1995)

De acordo com Wright (1995), o Net/3 é organizado em três camadas distintas, socket, protocolos e interface, e em três conjuntos de filas (Figura C.4). A camada de sockets oferece uma interface comum de chamadas de sistema, onde as aplicações podem utilizar os serviços de estabelecimento e encerramento de conexão, envio e recebimento de dados, tarefas de controle, etc. A camada de protocolos permite implementar diferentes famílias de protocolos, *Xerox Network Systems* (XNS), OSI, Unix e TCP/IP, entre outras, que, por sua vez, podem ser subdivididos em mais subcamadas. A camada de interface implementa os protocolos de controle de enlace e *drivers* específicos para os dispositivos de interface de rede, ou *Network Interface Cards* (NICs).

[20] A release 1.0 do Net/1 (BSD Networking Software Release 1.0) foi produzida em 1989, e a release 2.0, conhecida como Net/2, foi produzida em 1991. Em 1994 foi produzido o Net/3, também conhecido como BSD-Lite (WRIGHT, 1995).

C.2.1 Socket API

Um socket API é formado por uma estrutura de endereçamento de socket, que é passada em dois sentidos: a partir do processo usuário para o kernel do sistema operacional, e do kernel para o processo usuário. No segundo caso, por exemplo, o argumento resultante do valor da estrutura de endereçamento de socket é um ponteiro.

O processo usuário (aplicação) precisa informar o kernel sobre o tamanho da estrutura de dados (buffer) que será passada a ele. Dessa forma, quando a estrutura estiver cheia, o kernel não escreverá após o final da estrutura. Já o kernel precisa comunicar o processo usuário sobre a quantidade de informação que ele carrega atualmente em sua estrutura.

A Figura C.5(a) representa como a estrutura de endereçamento de socket é passada do processo usuário para o kernel do sistema operacional, sendo os argumentos da função um ponteiro para a estrutura de socket e um valor inteiro (que indica o tamanho da estrutura).

A Figura C.5(b) representa a passagem da estrutura de endereçamento de socket do kernel para o processo usuário, revertendo o sentido do cenário C.5(a). Nesse caso, os argumentos da função são dois ponteiros: um que aponta para a estrutura de endereçamento de socket e outro que indica o tamanho da estrutura. Esse tipo de argumento é visualizado na Figura C.5(b) em valor/resultado.

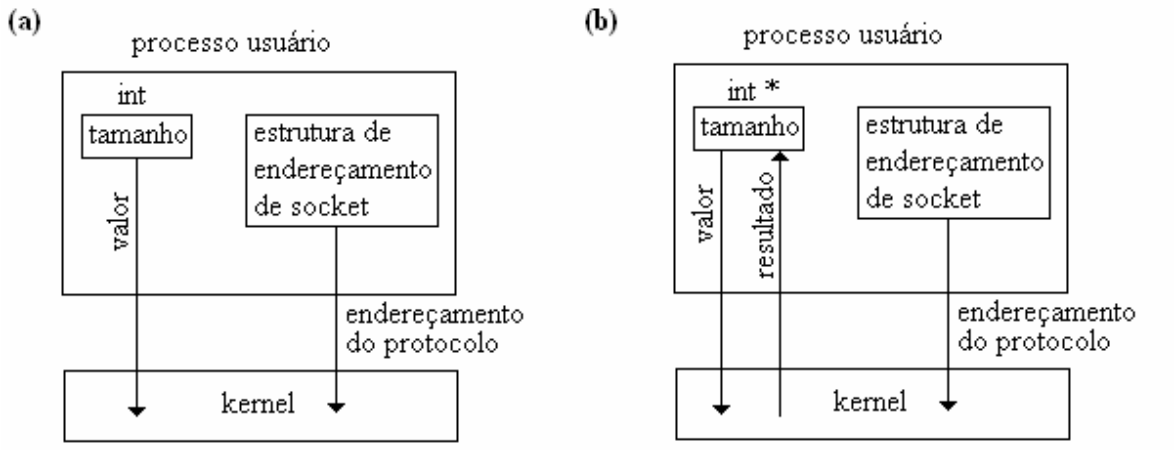


Figura C.5 – Estrutura de endereçamento de socket: (a) passagem do processo usuário ao kernel; (b) passagem do kernel ao processo usuário. Fonte: Adaptado de Stevens (1998).

C.2.2 Socket

Um socket é criado por meio de uma função socket (*sock_stream*), que retorna um pequeno descritor de arquivos (número inteiro), utilizado para identificar o socket em

chamadas futuras. Um par de sockets define dois pontos finais de uma conexão: endereço IP local, porta TCP local; e endereço IP remoto, porta TCP remota. Toda conexão TCP da Internet é identificada unicamente por um par de socket. Pode-se estender esse conceito ao UDP, ressaltando-se apenas que o UDP é um protocolo que fornece um serviço sem conexão.

Como exemplo demonstrativo sobre o funcionamento de socket, a Figura C.6 ilustra as funções básicas de um socket TCP, necessárias para estabelecer uma comunicação entre cliente e servidor, e o Quadro C.1 apresenta um resumo das descrições dessas funções. Mais detalhes de implementação podem ser encontrados em Stevens (1998).

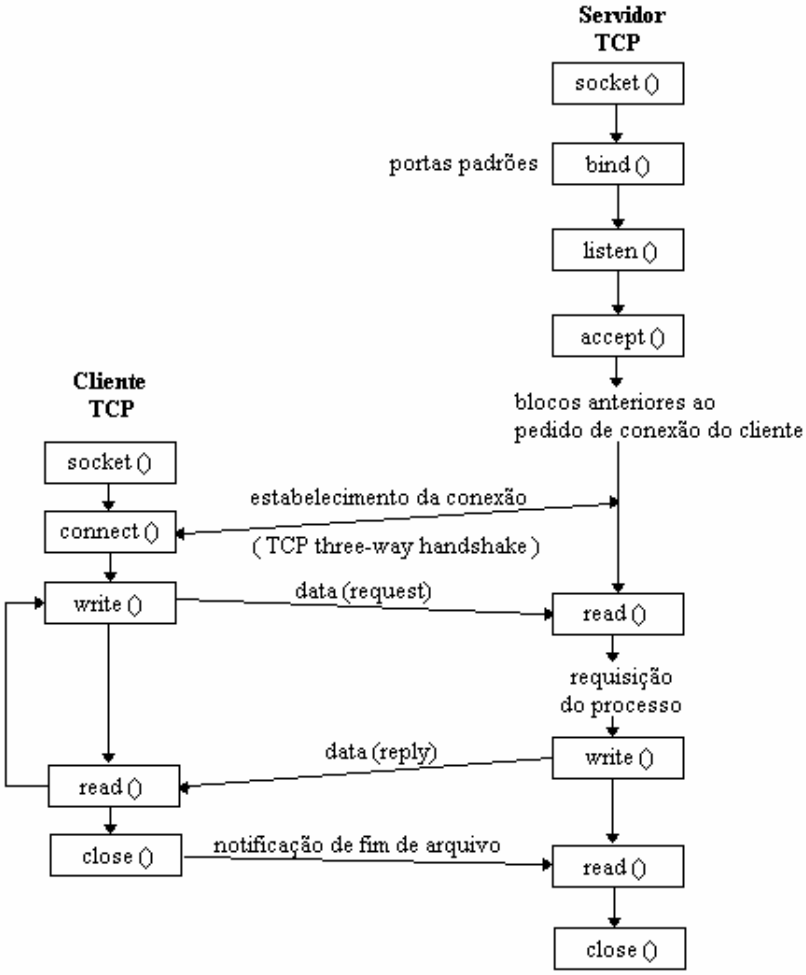


Figura C.6 – Funções básicas de um socket TCP cliente-servidor. Fonte: Adaptado de Stevens (1998)

FUNÇÃO	SINTAXE E DESCRIÇÃO
connect	#include <sys/socket.h> int connect (int *sockfd*, const struct sockaddr **servaddr*, socklen_t *addrlen*)[21]; Utilizada pelo cliente TCP para estabelecer uma conexão com o servidor TCP. Retorna o descritor de arquivo *sockfd*. O segundo argumento é um ponteiro para a estrutura de endereçamento do socket[22], e o terceiro é o tamanho da estrutura de endereçamento.
bind	#include <sys/socket.h> int bind (int *sockfd*, const struct sockaddr **myaddr*, socklen_t *addrlen*); Associa o endereço do protocolo local a um socket. Pode ser um endereçamento de 32-bit IPv4 ou 128-bit IPv6, junto com 16-bit de um número de portas TCP ou UDP. A chamada *bind* permite especificar um número de porta, um endereço IP, ambos ou nenhum.
listen	#include <sys/socket.h> int listen (int *sockfd*, int *backlog*); Chamada apenas pelo servidor TCP, executa duas ações. Quando um socket é criado pela função socket, ele é considerado um socket ativo. A função *listen* converte um socket não conectado em um socket passivo, indicando ao kernel que pode aceitar pedidos de conexões a esse socket. A segunda ação está relacionada ao número máximo de conexões que o kernel pode enfileirar para esse socket (argumento *backlog*).
accept	#include <sys/socket.h> int accept (int *sockfd*, struct sockaddr **cliaddr*, socklen_t *addrlen*);); É chamada pelo servidor TCP retornando a próxima conexão completada diante de uma fila de conexões completadas. Se a fila de conexões completadas estiver vazia, o processo é colocado para dormir. O segundo e terceiro argumentos são utilizados para retornar o endereço do protocolo de um processo par (cliente) conectado. Se a chamada *accept* teve sucesso, seu valor de retorno é uma marca de um novo descritor que foi automaticamente criado pelo kernel. Esse novo descritor referencia a conexão TCP com o cliente.

Quadro C.1 – Funções básicas de socket TCP

C.5 Considerações Finais

A arquitetura TCP/IP permite liberdade de implementação. Qualquer usuário da rede pode desenvolver aplicativos usando os padrões definidos ou criando seus próprios padrões. Isso pode ser considerado uma vantagem no âmbito de criação e agilidade no desenvolvimento de soluções; ou desvantagem, pois dificulta a integração, a organização e a transparência no funcionamento dos sistemas.

[21] Todas as funções retornam o valor 0 se OK e -1 se ocorrer erro.

[22] Definida na seção C.2.1 (Figura C.5).

> Uma implementação do conjunto do protocolo TCP/IP foi empacotada com o popular sistema operacional Unix, distribuído pela Universidade de Berkeley, na Califórnia, no início dos anos 1980. Pessoas estão satisfeitas em usar esse software que é mais vantajoso do que um software que requer esforço para adquirir, então, a disponibilidade de distribuição do Unix de Berkeley permitiu que a arquitetura TCP/IP alcançasse a massa crítica [...]. (PETERSON; DAVIE, 2000).

Este apêndice apresentou uma revisão sobre a infra-estrutura de comunicação da Internet, destacando alguns detalhes de implementações cujo entendimento é importante para constituir um modelo conceitual que se pretende adaptar à infra-estrutura da Internet.

AN EXTENDED MODEL FOR TCP/IP ARCHITECTURE

Dayna Maria Bortoluzzi
PPGEP - Federal University of Santa Catarina
The STELA group
dayna@led.br

Erivelto Souza Cunha
PPGCC - Federal University of Santa Catarina
ericunha@inf.ufsc.br

Elizabeth S. Specialski
3IT – Ilha Instituto Integrado de Tecnologia
beth@3it.org.br

ABSTRACT

This paper presents a proposal for the session layer introduction into the TCP/IP architecture, generating an extended model able to respond to actual and future application requirements. The main aspects of the proposed model as well as the advantages of its utilization are discussed. Considerations on the necessary efforts in order to perform the adaptations are also made.

1. INTRODUCTION

In the nineties, with the World Wide Web's success and the growing commercial explosion over the internet paradigm, the TCP/IP model consolidated its position as the most used architectural model on the web. Since then, applications are being loaded with new functionalities in order to respond to user's demands. New standardizations were created, mostly implemented as communication protocols in the application layer. As a result, nowadays there is an irreplaceable functional global network, composed by infinity extensions defined to respond to industrial, commercial and academic demands. These extensions were responsible for transforming a model, conceived as simple in its origins, into something complex and confusing.

Stronger models were developed aiming to create a network that would be able to comply with the application requirements. ATM is an example of a model that supports multimedia application [1]. However, any model of architecture has to consider the existing application demand, those that are used and consolidated by the internet. Taking the example of ATM's LAN-Emulation, it will always be necessary to develop an adaptation of an extended model to the TCP/IP architecture.

This work was motivated by the hypothesis that it is possible to reorganize the internet stack, in order to cover the diversity of existing and future application without expressively changing the present environment.

The following chapters present: the deficiencies and requirements of the current TCP/IP architecture, the proposal of an extended model able to answer the actual and future application requirements, the directions that can be taken in the implementation of this model and, finally the conclusion of the work.

2. DEFICIENCIES AND REQUIREMENTS PRESENTED BY THE CURRENT TCP/IP ARCHITECTURE

The internet architecture stands out for the simplicity of its protocols and for how efficiently it gets to interconnect heterogeneous systems.

When the TCP/IP architecture was implemented, very little was known about the current concepts of software engineering, which demand differentiation between specification and implementation. The model does not distinguish the concepts of service, interface and protocol, clearly enough. Although the implementation of the IP and TCP protocols were carefully projected, the same isn't true about the other protocols application layer. As a result, in the TCP/IP model, the protocols are very used, but the model is practically ignored [2].

Several of the protocols connected to the model execute similar tasks trying to respond to each application's specific requirements. for example, the "file transfer" task can be performed in several ways using the following protocols: FTP (file transfer protocol), SFTP (security file transfer protocol), or FTP associated with SCP (session control protocol), which allows to create various session connections over the same transport connection in which the idea is to transmit the data in different channels in order to accelerate the file transfer and guarantee the re-synchronism in case of a connection loss. Therefore, several functions concerning the session layer can be found in protocols such as: SSL, SCP, SIP and others.

An example of a session control mechanism, which is very used in web application, is the "cookie". It consists of a header inserted by the application that uses the HTTP protocol. The cookie performs the session control by storing information in personal computers in order to facilitate browsing.

Depending on the operational system or software used, it's possible to find differences in the implementation of "peer to peer" application. for example, initially the file transfer application uses port 21 for control and port 20 for data transferring, but nowadays some application use high ports randomly chosen (over 1024) for data transferring. Another characteristic to be considered is the "passive ftp", where the server determines the ports to being used for data transferring. When it comes to the "active ftp", the client chooses the ports to be used. It's possible to notice that there are several ways to perform a simple file transference and they should all be considered when using protection mechanisms such as package filters. A random choice of data transference ports or the use of

0-7803-8549-7/04/$20.00

active ftp compromises the security in package filters. Network administrators try to avoid this problem by using intermediate mechanisms (proxy) to re-direct TCP ports. This demands hardware and software resources, once these mechanisms are installed in the application layer and add more functionality to the system responsible for the package filter [3].

SIP (session initiation protocol) is an application-layer control protocol that can establish, modify, and terminate multimedia sessions (conferences) with one or more participant. These sessions can be internet telephone calls, distributed multimedia and multimedia conference [4]. SIP is a component that can be used to complement multimedia architecture, together with other protocols, such as RTP, RTSP, MEGACO and SDP. However, SIP basic functionality does not depend on any of these protocols. some solutions associate in SIP utilization at networks with NAT (network address translator) are available in drafts, for example, the problem of sending information about IP address and transport ports of RTSP protocol (real-time streaming protocol), when the media flow is blocked because of NAT [5]. In this situation, the problem occurs in network layer and, again, the solution is done in application layer.

The great explosion of multimedia application is a fact to be considered. The same way in the 90's the WWW was responsible for the internet explosion, nowadays, there is a great number of users who connect to the web looking for the multimedia world. many multimedia application generate outputs on constant bits rates (CBR - constant bit rate), some examples are the non-compacted audio streams, which as a telephones quality that generate a constant rate of 64kbps (8000samples/s of 8 bits) and also the real time application that involve CBR data stream. In IP (internet protocol) networks, this kind of traffic is hard to be carried on, since most of the networks were not prepared for such task. Application designers solve this problem by including synchronization, buffering and sometimes retransmission mechanisms. The difficulty identified in these cases is that each application needs to solve the problem and does it in an individual form, because there is no a common support infrastructure.

Based on these considerations, it's noticeable that an intermediate layer is necessary. It should be placed between the application layer and the transport layer, as a way to make up for the application requirements, avoiding common facilities replication and guaranteeing the interoperability of the existing systems. Moreover, the introduction of this layer, responsible for executing tasks related to the session layer, should preserve the model's layers independence concept.

3. AN EXTENDED MODEL FOR TCP/IP ARCHITECTURE

The communication hierarchic model proposed by OSI distinguishes two different layer classes: the upper layer, represented by layers 5 to 7, which services are oriented to application written by users, with the purpose of simplifying their tasks by supplying diverse standardized services; and the lower four layers, represented by layers 1 to 4, which services are oriented to the information transport, dealing essentially with communication problems such as: sending data, waiting for acknowledgment, sequencing data that arrives out of order, calculating and verifying checksums, and so on [6].

The session layer is the first one (from bottom up) positioned in the second class. This is actually, one of the simplest layers of the OSI model, offering a limited amount of services, far from the services offered by layers such as the application and transport.

Opposite from OSI, in the TCP/IP model there aren't session and presentation layers; the application layer directly accesses the transport layer service primitives. Services offered by OSI layers 5 and 6, are implemented by several application layer protocols.

Aiming to organize the internet stack, a model was developed and it suggests the separation of session control functionalities in an independent layer. Following the TCP/IP logic, the proposed session layer uses the transport layer services and provides other services for the application layer. The figure 1, showed bellow, allows visualizing the extended model.

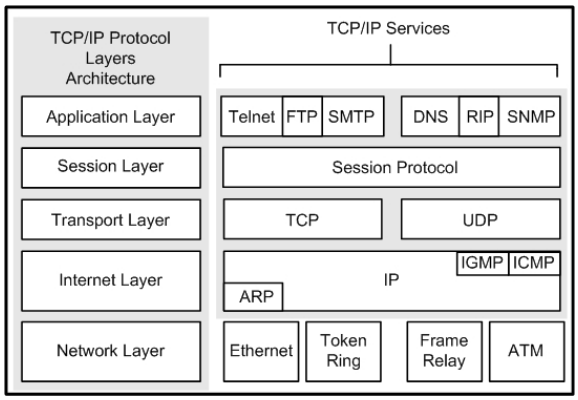


Figure 1 – an extended model for TCP/IP architecture

The proposed session layer gets basic concepts from OSI reference model and evaluates the requirements and implementations of current internet application. the existence of a number of application not requiring additional session services due they're working perfectly in the present environment, was also considered.

4. THE TCP/IP SESSION LAYER

The solution proposed consists in separating the session control functionality, whether from the application protocols or the transport protocols, with the modeling and implementation of a middleware. Two researches stages were considered: the session layer OSI functions and the session protocols for internet applications.

4.1. Services provide by the session layer

Among the services defined in the OSI model session layer, synchronizing services, dialog management, activity management such as services included in part of the application, were identified [7,8]. Besides the identification of these services, it was necessary to consider an application might not use any of the facility offered by the session layer, directly accessing the TCP or UDP transport instead. Therefore, the pass-through service was included. Complementarily, connection

establishment and release services, regular data transference, typed data transference, and expedited (urgent) data transference were included, with the same functionality defined in the OSI model session layer.

Three different relationships between a session connection and a transport connection were considered: a session connection may use a single transport connection or many transport connections in data transmitting and, a transport connection may be related with one or more session connections [9].

The use of any service offered by the session layer must be negotiated between the peers (users) to set values of various parameters of a session connection. An exception is the pass-through service.

The management dialog is implemented in a simplified way. When an application requires the data transference to be done as a half-duplex, it must negotiate the use of the token during the connection establishment.

The synchronizing mechanism was also simplified so that only one type of synchronism point will be placed as an information border. The synchronism point placement must be explicitly recognized by the receptor.

Another important session layer function is the activity management. This task is based on the concept of data stream decomposing in activities, independently of each other. The activity concept will depend on the application considered, and the user is responsible for this definition. There are several activities to be executed by the session layer. Only by the end of each activity the application will execute a task or not. The purpose of defining activities is the fact that certain application might have atomicity guarantee, avoiding errors due to failures occurred during the actions execution of a same activity.

4.2. Session protocol elements

The session service is implemented by a session protocol that is used between two session entities. When defining a session protocol for the TCP/IP architecture, the development of 4 services to be implemented in the session entities is proposed:

- Connection oriented service: uses the TCP services provided by the transport layer
- Connectionless service: uses the UDP services provided by the transport layer
- TCP pass-through: does not implement session service, merely offers a direct passage interface from the application to the TCP protocol of the transport layer
- UDP pass-through: does not implement session service, merely offers a direct passage interface from the application to the UDP protocol of the transport layer.

The pass-through services are defined to serve applications that have already implemented its own session control, in order to facilitate the application adaptation to the proposed model.

The definition of a confirmed session service with a connection, aims to respond to the demand of application that require session control and use the TCP protocol as a transport mechanism.

Besides the data transfer, the proposed model presents a synchronism service. The utilization of synchronism mechanism over connectionless mode aims to respond to the demand of multimedia application. In this case, application are divided in two groups: off line application, which retransmission from synchronism points is valid, and on line application (in real time), which retransmission is not important, but in this case, quality of services reports can be generated, from the synchronism points and forwarded to the application.

4.3. Session establishment

This phase is based on a session connection establishment between two session layer users and on parameters negotiations that will define services to be used during the connection.

When the customer initializes a session, parameters are changed in order to begin the session services. Parameters such as id_protocol, id_category and id_instance, defined in the session control header are known by both the client and the server, while the id_session is randomly generated and must be a single value, non-duplicated, once it will be used in the data synchronization function.

4.4. Data transference

Three services related to data transference are defined:

- Regular data transfer, which allows transferring regular data unities during the session connection.
- Expedited data transfer, which allows sending limited size data, in a connection free from control limitations. They are normally used to send control information when data transport is restricted.
- Typed data transfer is like regular data, except that it may be sent without regard to the ownership of any tokens.

4.5. Dialog management

Dialog management allows only the user holding the token to send data; the other one must remain silent. When the user holding the token has finished transmitting, it passes the token to the other one. in our model, the token only makes sense in a half-duplex mode. The full duplex option is not considered. This dialog mechanism is useful to multimedia application, like videoconference.

4.6. Synchronism points

The synchronization point allows a constant data (regular, expedited, typed) stream transmission defined by dialog unities.

Resynchronization is used so that the connection can be reestablished in a former synchronization point, or in a new one. The data sent after the last synchronization point might be lost.

Once the server gets the customer information from a synchronism point, the next point can never be lower than a point previously confirmed in a certain session identifier. For security reasons, the one that was already transferred and confirmed can’t be transferred again.

4.7. Activity management

To make use of this concept the application layer user must to indicate the beginning of a new activity through id_ativ. From this moment on, only the OSI primitives will be considered:

- Activity start: for the user to explicitly indicate that a new activity is beginning;

- Activity resume: indicates that an activity previously interrupted is being restarted. It uses a session identifier to resume the activity;
- Activity interrupt: abnormally stops an activity which data storage had already been confirmed (via synchronization point), so that they won't be retransmitted when the connections is reestablished. The data sent after the last confirmed synchronization point is lost;
- Activity discard also provokes an abnormal activity stop, but in this case all the data send since the beginning of the activity are discarded and the activity cannot be resumed;
- Activity end indicates the end of the started activity; at the same time it sends a main synchronizing point.

4.8. Retransmission

The session layer offers both, guaranties in case of connection loss and in case of an application restart. Its function is to keep the synchronism, no matter what fails in the data transmission.

A session is restarted if the id_session is recognized. The customer doe not keep id_session cache; only the provider stores these data whenever TTL is higher than zero. When a session is restarted, the parameters (IP, port, id_protocol, id_category, id_instance) must be transferred from the client to the server that will calculate the id_session value, to locate the other session control parameters. When the renegotiation is over, both the client and the server have the session information loaded.

The servers must anticipate the TCP ports sequential substitution, and that in networks in which the DHCP is used, the IP address might also change. It's unsafe to allow re-initialization of a different IP address session, because messages coming from a determined IP can be easily forged [10].

4.9. Session ending

The model allows the connection to be normally ended by one of the communicating parts, users or provider unities, which implies data loss.

5. CONSIDERATIONS ON IMPLEMENTATION

To validate the model, we are considering two options for the session layer implementation: one on the user level that can be seen in option (a) of the figure 2, and another one, on the operational system kernel level, shown in option (b). In implementations on the user level there is more flexibility, making it possible to optimize the session control service for a specific base application, losing in performance, though. In implementations on the operational system kernel level, there is a gain in performance.

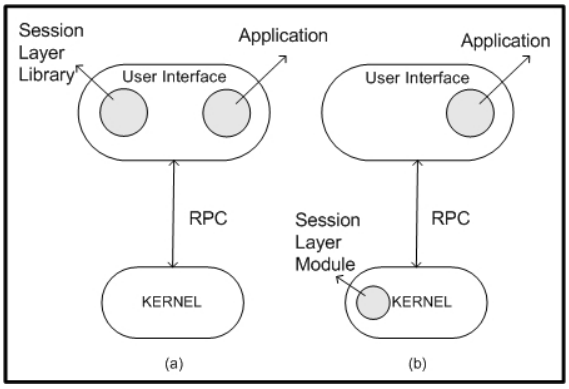


Figure 2 – Session layer implementation

According to [11] it's possible to have network protocol software executing in a user level, with a better or at least the same performance as when executing in the operational system kernel level. This implementation's advantage is the flexibility because the code is separate from the kernel, allowing it to be easily modified and optimized. Following this alternative, we would adopt the strategy of implementing the four session services as compiled libraries in each application's addressing space. This alternative might have an implementation performance advantage in the kernel level and the flexibility as well as the portability of a user's level application.

A second option is to implement the session layer in Linux's operational system kernel the same way the transport layer implementation is done. Through this alternative we would alter the kernel, placing the session layer right after the BSD sockets, so that the application wouldn't access the transport layers without passing through the session layer. It's possible to visualize the implementation of the transport and network layer in the Linux system kernel in figure 3.

In Linux, as well as in Unix BSD, the sockets offer a mechanism used for communication inter-processes, normally using the customer-provider architecture. Such processes don't necessary need to be in the same machine, a remote transport connection can be open and the messages transmitted have no size limit. in order to guarantee the messages delivery in the destination machine, the communication protocols are implemented right bellow the sockets interface

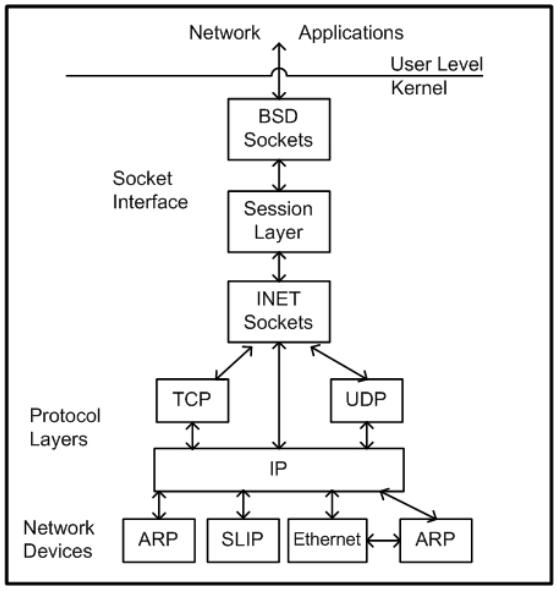


Figure 3 – Session layer location in the Linux kernel

6. CONCLUSION

The proposed model allows the integration between current applications and maintains the integrity of what has already been developed and is working. It is able to respond to most of session control present requirements as well as future expansion. The main objective is to offer a simplified and functional session service, able to respond to existing application demands, at the same time it seems appealing to programmers.

Therefore, there is an expectation that new versions of these applications will be made using this session protocol that will gradually contribute to the current internet stack.

An internet application mapping is under development, identifying requirements and services related to the session layer. Besides, we are adapting some applications, in order to use the proposed model. Specific functionalities will be implemented as an extension to the proposed model.

This study is part of a doctorate thesis and the data related to the protocol header and the services primitives can be altered up to the completion of our work. We also intend to give some contribution to the model that could aid with future application requirements.

6. BIBLIOGRAPHY

[1] W. Stallings, “High-Speed Networks TCP/IP and ATM Design Principles,” ISBN 0-13-525965-7 New Jersey: Prentice-Hall, 1998.

[2] A. S. Tanenbaum, “Computer Networks,” ISBN 0-13-349945-6 New Jersey: Prentice-Hall, Third Ed.1996.

[3] S. Bellovin, “Firewall-Friendly FTP,” Request for Comments 1579, February 1994.

[4] J. Rosenberg; H. Schulzrinne; G.Camarillo; A. Johnston; J.Peterson; R.Sparks; M.Ha'ndley; E.Schooler, “SIP: Session Initiation Protocol”, Request for Comments 3261, June 2002.

[5] M. Westerlung, “How to make Real-Time Streaming Protocol (RTSP) traverse Network Address Translators (NAT) and interact with Firewalls,” draft-ietf, February 2003.

[6] W. R. Stevens, “Unix Network Programming: Network APIs – Sockets and XTI,” ISBN 0-13-490012-X, New Jersey: Prentice-Hall, Second Ed. 1998.

[7] F. Halsall, “Data Communications, Computer Networks and Open Systems”. ISBN 0-201-42293-X, Harlow, England, Addison-Wesley, Fourth Ed. 1997.

[8] ITU-T, X.215: Information Technology - Open System Interconection – Session Service Definition. ITU-T, 1995.

[9] ITU-T, X225: Information Technology - Open System Interconnection – Connection-Oriented Session Protocol: Protocol Specification. ITU-T, 1995.

[10] E. Rescorla, “SSL and TLS Designing and Building Secure Systems,” ISBN 0-201-61598-3. Addison-Wesley, August 2001.

[11] C. Maeda; B.N Bershad, “Protocol Service Decomposition for High-Performance Net-working”, ACM Symposium on Operating Systems Principles, pp 244-255, 1993.